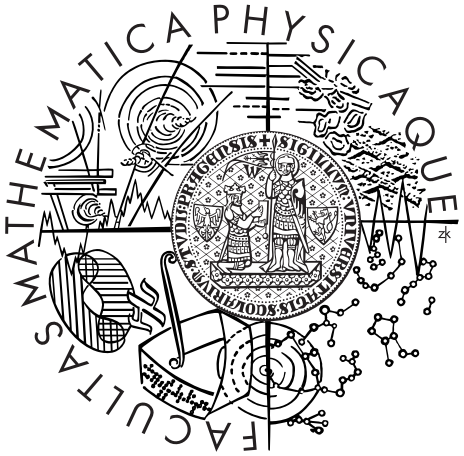
Charles University in Prague
Faculty of Mathematics and Physics

DOCTORAL THESIS



Ing. Jiří Novák

Similarity Search in Mass Spectra Databases

Department of Software Engineering

Supervisor of the doctoral thesis: doc. RNDr. Tomáš Skopal, Ph.D.

Study programme: Computer Science

Specialization: Software Systems

Prague 2013

I would like to thank my supervisor doc. RNDr. Tomáš Skopal, Ph.D. and my consultant RNDr. David Hoksza, Ph.D. for many valuable advices during my Ph.D. study. I would like to also express deep gratitude to my family for having the patience with me, especially to my wife MUDr. Lucia Nováková, my funny son Jiří, and my parents Miroslava Nováková and Jiří Novák.

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In Prague date April 22^{nd}, 2013 Signature of the author

Annotation

Title:

Similarity Search in Mass Spectra Databases

Author:

Ing. Jiří Novák
email: novak@ksi.mff.cuni.cz

Department:

Department of Software Engineering
Faculty of Mathematics and Physics
Charles University in Prague

Supervisor:

doc. RNDr. Tomáš Skopal, Ph.D.
email: skopal@ksi.mff.cuni.cz

Abstract:

Shotgun proteomics is a widely known technique for identification of protein and peptide sequences from an "in vitro" sample. A tandem mass spectrometer generates tens of thousands of mass spectra which must be annotated with peptide sequences. For this purpose, the similarity search in a database of theoretical spectra generated from a database of known protein sequences can be utilized. Since the sizes of databases grow rapidly in recent years, there is a demand for utilization of various database indexing techniques. We investigate the capabilities of (non)metric access methods as the database indexing techniques for fast and approximate similarity retrieval in mass spectra databases. We show that the method for peptide sequences identification is more than 100× faster than a sequential scan over the entire database while more than 90% of spectra are correctly annotated with peptide sequences. Since the method is currently suitable for small mixtures of proteins, we also utilize a precursor mass filter as the database indexing technique for complex mixtures of proteins. The precursor mass filter followed by ranking of spectra by a modification of the parametrized Hausdorff distance outperforms state-of-the-art tools in the number of identified peptide sequences and the speed of search. The proposed methods are implemented in the peptide identification engine SimTandem which can be used for a batch analysis in the framework TOPP based on OpenMS.

Keywords:

tandem mass spectrometry, peptide identification, metric and non-metric access methods, similarity search, bioinformatics

Anotace

Název práce:

Podobnostní vyhledávání v databázích hmotnostních spekter

Autor:

Ing. Jiří Novák
email: novak@ksi.mff.cuni.cz

Katedra:

Katedra softwarového inženýrství
Matematicko-fyzikální fakulta
Univerzita Karlova v Praze

Školitel:

doc. RNDr. Tomáš Skopal, Ph.D.
email: skopal@ksi.mff.cuni.cz

Abstrakt:

Tandemová hmotnostní spektrometrie je známá metoda pro identifikaci proteinových a peptidových sekvencí ze vzorků biologického materiálu. Hmotnostní spektrometr generuje desetitisíce spekter, která musí být následně anotována peptidovými sekvencemi. Za tímto účelem lze využít podobnostní vyhledávání v databázích teoretických spekter generovaných z databází známých proteinových sekvencí. Vzhledem k tomu, že objem těchto databází každoročně narůstá téměř exponenciálním tempem, je zapotřebí hledat nové způsoby pro jejich indexování. V této práci se zaměřujeme na využití (ne)metrických přístupových metod jako databázových indexů pro rychlé a aproximativní podobnostní vyhledávání v databázích spekter. Navržená metoda identifikace peptidových sekvencí dosahuje více než 100-násobného zrychlení oproti sekvenčnímu průchodu celé databáze, přičemž je správně anotováno přes 90% spekter. V současnosti je metoda vhodná zejména pro malé směsi proteinů. Pro komplexní směsi proteinů využíváme indexovací metodu založenou na prekurzorovém hmotnostním filtru, která má při použití s modifikací parametrizované Hausdorffovy vzdálenosti vyšší rychlost i přesnost vyhledávání než běžně používané metody. Navržené metody jsou implementovány v aplikaci SimTandem, kterou lze použít pro dávkové zpracování ve frameworku TOPP založeném na knihovně OpenMS.

Klíčová slova:

tandemová hmotnostní spektrometrie, identifikace peptidů, metrické a nemetrické přístupové metody, podobnostní vyhledávání, bioinformatika

Contents

Preface

Proteins are the basis of all living organisms while the tandem mass spectrometry is a widely used technique for identification and quantification of peptides and proteins from an "in vitro" sample. A mass spectrometer produces tens of thousands of mass spectra which must be annotated with peptide sequences. One of the commonly used approaches for annotation of spectra is the similarity search in a database of theoretical spectra generated from a database of known protein sequences. Since the sizes of databases grow rapidly in recent years, there is a demand for fast similarity search in these databases. A way how to speed-up the search in a database is to utilize an indexing technique. The (non)metric access methods are database indexing techniques which are based on properties of (non)metric spaces and they are suitable for various kinds of multimedia data.

Summary of Contributions

In this thesis, we investigate the utilization of database indexing techniques for mass spectra databases. We map the recently proposed techniques and analyze the capabilities of (non)metric access methods for fast and approximate similarity search in mass spectra databases. The proposed method has been successfully tested on small mixtures of purified proteins (up to tens of proteins). However, due to a rapid development of new and very accurate instruments, the utilization of (non)metric access methods on complex mixtures of proteins (containing thousands of proteins) is a non-trivial task. From this reason, we also investigate the utilization of a precursor mass filter followed by a ranking of theoretical spectra by a modification of the parameterized Hausdorff distance originally designed for (non)metric indexes. We show that the method outperforms state-of-the art tools on complex mixtures of proteins in both – the number of identified peptide sequences and the speed of the search. The proposed algorithms have been implemented in the application SimTandem which can be used for a batch analysis in the framework TOPP based on OpenMS.

Structure of Thesis

The thesis is organized as follows. In Chap. 1, proteins are briefly introduced and an overview of amino acids is proposed. In Chap. 2, the basic physical and chemical principles of mass spectrometry are described. The Chap. 3 gives an overview of existing algorithmic techniques for identification and quantification of peptides/proteins from databases of theoretical mass spectra, and for statistical

evaluation of peptide-spectrum matches. Since the thesis is focused on the indexing of mass spectra, the Chap. 4 maps the existing approaches which speed-up the search in mass spectra databases. In Chap. 5, the metric and non-metric access methods are described. The approach for identification of peptides by non-metric access methods is proposed in Chap. 6. In experimental Chap. 7, the efficiency and effectiveness of the algorithm for identification of peptide sequences by non-metric access methods are studied, and a statistical comparison of precursor mass filter with state-of-the art tools is proposed. In Chap. 8, the implementation of SimTandem is described.

Chapter 1

Introduction

The genetic information of prokaryotes and eukaryotes is encoded in DNA (deoxyribonucleic acid) which determines their characteristics. The DNA is organized as a double helix having two complementary strands. A strand of the helix can be represented by a linear sequence over the alphabet of four bases – *adenine*, *cytosine*, *guanine* and *thymine* (A, C, G and T). While adenines in one strand are paired with thymines in the other strand, the cytosines are coupled with guanines. A triplet of bases (or codon) encodes an amino acid. Even though there are $4^3 = 64$ different triplets of bases, they encode only 20 amino acids because some amino acids are encoded by more triplets and because some triplets serve as stop codons. As defined by the *central dogma* of molecular biology, the DNA is transcripted into RNA (ribonucleic acid) and the RNA is translated into proteins.

Proteins are linear chains of amino acids which are connected by peptide bonds (Fig. 2.7). Proteins are the basis of all living organisms and they are essential for correct construction of cells and for their proper functioning [1]. In terms of computer science, a protein can be understood as a linear sequence over 20-letter subset of the English alphabet where each letter corresponds to an amino acid.

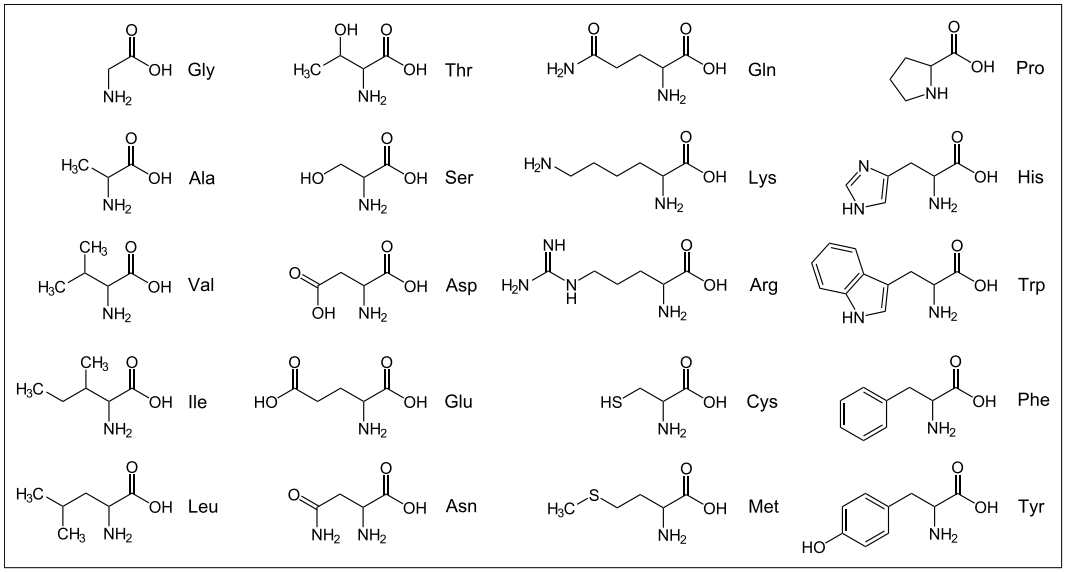


Figure 1.1: Basic types of amino acids

An amino acid is a chemical compound made from an amino group (-NH_2) and a carboxyl group (-COOH). The groups are connected to a carbon called C_α. Even though the basic structure of all amino acids is same, the amino acids differ by *side chains* (-R) connected to C_α. The side chains determine various physicochemical properties of amino acids (e.g., hydropathy, volume or polarity). The basic types of amino acids are shown in Fig. 1.1, their abbreviations and relative masses are proposed in Tab. 1.1. Masses of amino acids are given in Daltons (Da), what are relative molecular mass units. A *monoisotopic* mass is a sum of masses of atoms where the masses of atoms correspond to the masses of the most abundant isotopes occurring in the nature. An *average* mass is a sum of masses of atoms where the masses of atoms are weighted according to the occurrence of different isotopic forms of these atoms in the nature. Masses of amino acids are commonly given as residue masses, i.e., without the masses of terminal groups -H and -OH, which are substituted by peptide bonds when amino acids are chained in a protein.

The protein sequence is a *primary* structure of a protein. However, we also distinguish the *secondary*, *tertiary* and *quaternary* protein structure. The secondary structure is a 3D local segment of a protein (e.g., α-helix or β-sheet). The tertiary structure is a 3D structure of a protein and the quaternary structure is a 3D complex of multiple proteins connected by weak chemical interactions. Basically, a function of a protein is derived from its 3D structure while the protein sequence determines the protein structure. However, the similarity between protein sequences does not imply the similarity between protein structures. The reason is that the evolution tends to preserve the structure (i.e., the function) of a protein rather than the sequence.

Protein sequences are determined from "in vitro" protein samples, while tandem mass spectrometry is a fast and popular method for this task (Chap. 2). Known protein sequences are collected in freely available protein sequence databases MSDB [2], UniProtKB [3], NCBI [4], etc.

Amino acid	**Abbreviations**		**Residue mass [Da]**	
			Monoisotopic	**Average**
Alanine	Ala	A	71.03712	71.08
Arginine	Arg	R	156.10112	156.2
Asparagine	Asn	N	114.04293	114.1
Aspartic acid	Asp	D	115.02695	115.1
Cysteine	Cys	C	103.00919	103.1
Glutamine	Gln	Q	128.05858	128.1
Glutamic acid	Glu	E	129.0426	129.1
Glycine	Gly	G	57.02147	57.05
Histidine	His	H	137.05891	137.1
Isoleucine	Ile	I	113.08407	113.2
Leucine	Leu	L	113.08407	113.2
Lysine	Lys	K	128.09497	128.2
Methionine	Met	M	131.04049	131.2
Phenylalanine	Phe	F	147.06842	147.2
Proline	Pro	P	97.05277	97.12
Serine	Ser	S	87.03203	87.08
Threonine	Thr	T	101.04768	101.1
Tryptophan	Trp	W	186.07932	186.2
Tyrosine	Tyr	Y	163.06333	163.2
Valine	Val	V	99.06842	99.07

Table 1.1: Residue masses and abbreviations of amino acids

Chapter 2

Mass Spectrometry Fundamentals

Tandem mass spectrometry combined with high-pressure liquid chromatography (HPLC-MS/MS) is a widely used technique for identification and quantification of proteins and peptides [5] [6] [7] [8].

We can easily analyze small mixtures of purified proteins as well as complex mixtures of proteins obtained by a cell lysis containing thousands of proteins. A tandem mass spectrometer commonly generates tens of thousands of mass spectra corresponding to peptides (i.e., small pieces of proteins). In this chapter, we briefly describe the basic physical and chemical principles of mass spectrometry. The understanding of these principles is important for further computational processing of mass spectra.

2.1 Protein Digestion

In the bottom-up proteomics (Sec. 3.3.1), the proteins are commonly digested into peptides by an enzyme before the analysis by a spectrometer. The most common and cheap enzyme is trypsin which splits proteins after each amino acid lysine (K) and arginine (R) if they are not followed by proline (P) [9] [10]. Since the digestion is not perfect in practice, the tools for identification of peptides commonly allow to set up a *maximum number of missed cleavage sites*. For example, assume a peptide sequence "GHPETLEKFDK". Since the peptide is not digested after the first "K", the number of missed cleavage sites is equal to 1. An overview of digestion enzymes is presented in Tab. 2.1 [11].

2.2 High-pressure Liquid Chromatography

In general, the chromatography is a technique for separation of mixtures. A mixture is dissolved in a liquid called the *mobile phase* which carries the mixture through an immobilized porous substance denoted the *stationary phase*. The high-pressure liquid chromatography (HPLC) is a technique which is commonly used in proteomics for separation of peptides before the analysis by a mass spectrometer. The mobile phase passes through a column and carries separated peptides out of the column [5]. Originally, the liquid chromatography (LC) used

Enzyme	Cleaves at:	Except if:
arg C	after R	before P
asp N	before D	
chymotrypsin	after F, (L, M,) W or Y	before P; after PY
cyanogen bromide	after M	
Glu C (basic)	after E	before P or E
Glu C (acidic)	after D or E	before D or E
Lys C	after K	
pepsin (high acidity)	after F or L	
pepsin (low acidity)	after A, E, F, L, Q, W or Y	
proteinase K	after A, C, F, G, M, S, W or Y	
trypsin	after K or R	before P

Table 2.1: Protein digestion enzymes

the gravity force to pass the mobile phase through the stationary phase. However, in modern HPLC techniques, high-pressure pumps are used to get reasonable flow rates.

Commonly, a detector is placed at the outlet of the column which detects components as they pass out of the column. Then a *chromatogram* is captured by an interconnected computer. The chromatogram is a 2D graph, having *retention time* along the horizontal axis and the intensities of eluted components along the vertical axis. The retention time determines when specified peptides elute from the column. Since the retention time can be also predicted from peptide sequences (usually, by methods based on the machine learning), it can be used as an auxiliary information when mass spectra are being annotated with peptide sequences [12].

2.3 Mass Spectrometry

In principle, a mass spectrometer consists of three parts – an *ion source*, a *mass analyzer* and a *detector*. The ion source charges neutral molecules which become ions. The mass analyzer separates charged ions by $\frac{m}{z}$ ratios where m is the mass of a ion and z is the charge of the ion. The $\frac{m}{z}$ ratio can be calculated as shown in Eq. 2.1 [13], where M_r is the molecular relative mass of a neutral molecule and $A_r(H) = 1.00794$ is the relative atom mass of the hydrogen. The detector measures the intensities of ions with specific $\frac{m}{z}$ ratios and forms *mass spectra*.

$$\frac{m}{z} = \frac{M_r + zA_r(H)}{|z|} \tag{2.1}$$

2.3.1 Ionization Techniques

Since mass analyzers are unable to detect neural molecules, the molecules must be charged by an ion source before the mass analysis. In practice, two main ionization techniques are utilized for biomolecules – *MALDI* and *ESI* [14]. These techniques are also known as *soft* because ions are not fragmented during the ionization.

MALDI

When the Matrix Assisted Laser Desorption Ionization (MALDI) is utilized, the analyte (proteins or peptides) is dissolved and mixed with a matrix. Then the mixture is crystallized at a MALDI plate (Fig. 2.1). The molecules of matrix are bombarded by short laser beams and molecules of analyte are ionized by protons transferred from the matrix. The absorbed energy causes that the molecules of matrix and analyte are ejected from the plate. MALDI generates ions with charges $z = 1^+$ and works under the vacuum or very low pressure. It is the dominating ionization source in single mass spectrometry (Sec. 2.4.1) and is commonly used with TOF mass analyzers (Sec. 2.3.2)

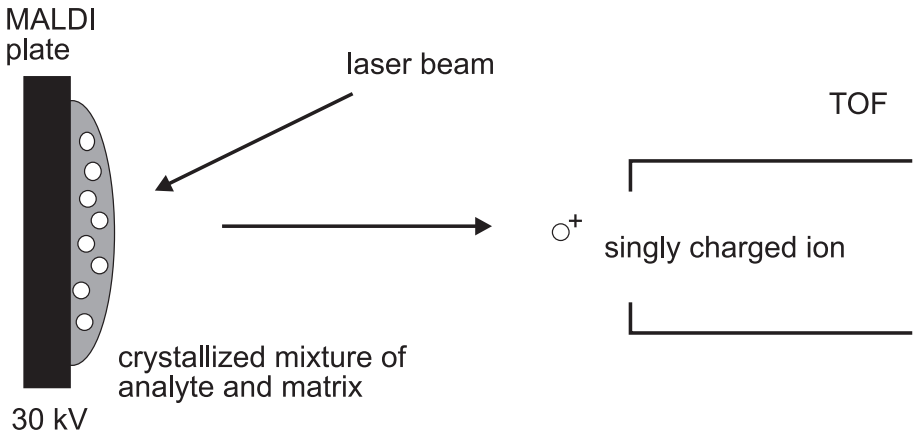


Figure 2.1: Principle of MALDI

ESI

The Electrospray Ionization (ESI) is able to work under atmospheric pressure and it is likely the most dominating ionization technique for tandem mass spectrometry in the combination with HPLC (Sec. 2.4). The dissolved molecules of

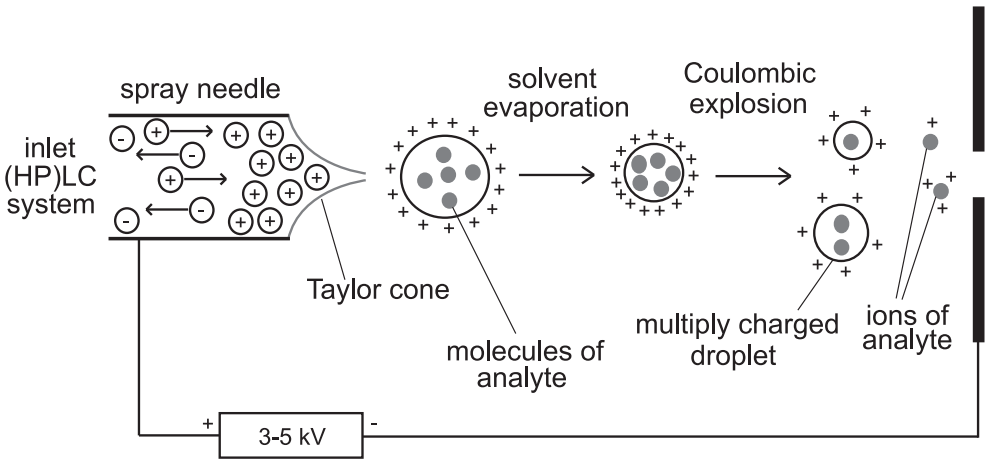


Figure 2.2: Principle of ESI

analyte are brought through a spray needle into the ionization source (Fig. 2.2). In the ion source, small droplets are arising because of a torrent of nitrogen. The droplets carry many charges what is caused by high voltage in the spray needle (3-5 kV). The evaporation of solvent causes that droplets become smaller and that electrostatic charge densities are higher. When a critical density is reached, a droplet is broken into smaller droplets. This effect is known as the *Coulombic explosion*. The Coulombic explosions are repeated until charged ions are released from droplets.

The released ions are commonly multiply charged what is advantageous for tandem mass spectrometry because both types of fragment ions can be captured when a multiply charged peptide ion (i.e., $z \geq 2^+$) is being fragmented (Fig. 2.7). In top-down proteomics (Sec. 3.3.2), the ESI enables the identification of big proteins because high charges cause that these proteins can be detected by a mass analyzer. For example, assume a mass analyzer having the range of measured $\frac{m}{z}$ values up to 3,000 Da. When a protein ion having mass 10,000 Da carries the charge 4^+, it generates the $\frac{m}{z}$ value of 2,501 Da (Eq. 2.1) and thus the ion is in the range of the mass analyzer. On the other hand, the multiply charged ions commonly complicate the identification of proteins because a mass spectrum contains more peaks having different charges but corresponding to the same molecule. Thus a *deconvolution* of spectra must performed which detects and eliminates these peaks.

2.3.2 Mass Analyzers

A mass analyzer is the main part of a mass spectrometer which separates charged ions by $\frac{m}{z}$ values [5] [14] [13]. The analyzers are based on different physical principles. We briefly describe several most common types of analyzers.

TOF

The time-of-flight (TOF) analyzer uses an electric field to accelerate ions through a drift tube. The time t is measured for which the ions reach a detector. The speed of ions vary according to their masses. Simply said, light ions reach the detector earlier than heavy ions. When an ion is accelerated into the drift tube, its potential energy is converted to kinetic energy (Eq. 2.2). The potential of the acceleration field is P, and e is the charge of an electron. The ion has a velocity v.

$$zeP = \frac{1}{2}mv^2 \tag{2.2}$$

The time for the ion to reach the detector is $t = \frac{d}{v}$, where d is the distance to the detector. Thus we can substitute v to obtain the Eq. 2.3.

$$t^2 = \frac{d^2}{v^2} = \frac{m}{z}\frac{d^2}{2eP} \tag{2.3}$$

We can observe that the $\frac{m}{z}$ value of a ion can be calculated from the time of flight (Eq. 2.4), where C is a constant.

$$t = \frac{d}{\sqrt{2eP}}\sqrt{\frac{m}{z}} = C\sqrt{\frac{m}{z}} \tag{2.4}$$

Quadrupole

Another common mass analyzer is the quadrupole formed from four parallel metabolic rods which are passed through by ions (Fig. 2.3) [15]. The quadrupole analyzer stabilizes paths of ions with a specific $\frac{m}{z}$ value using oscillating electrical field while the other ions collide with the rods. By changing the oscillation frequency, ions with all $\frac{m}{z}$ values can be passed through the rods and a mass spectrum is captured.

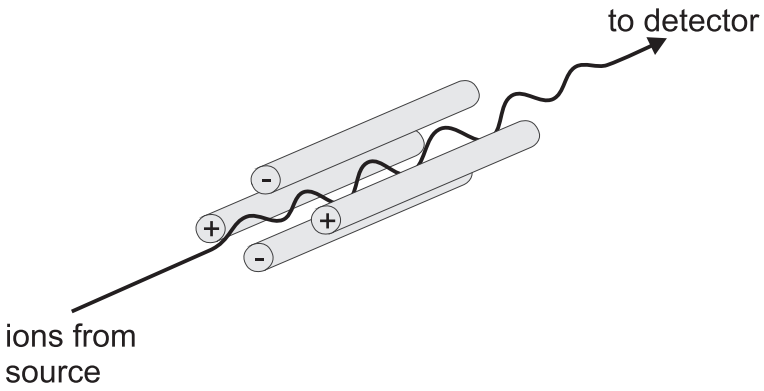


Figure 2.3: Principle of quadrupole

Ion Trap

Ion traps are based on the same physical principles like the quadrupole with the difference that all ions are trapped and only ions with a specific $\frac{m}{z}$ value are ejected sequentially. The basic kinds of ion traps are 3D ion trap, linear ion trap and orbitrap [5]. For example, the orbitrap (Fig. 2.4) [16] consists of an inner and an outer electrode which form an electrostatic field. The ions perform an orbitally harmonic oscillation along the axis of the electrostatic field. The frequency of oscillation is inversely proportional to $\frac{m}{z}$ values of ions. Orbitraps have high accuracies and they are patented by Thermo Scientific [17].

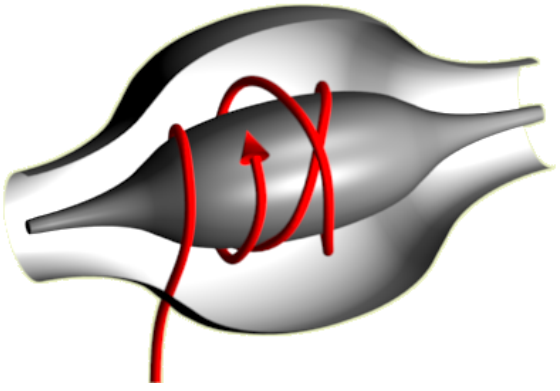


Figure 2.4: Principle of orbitrap

2.4 Tandem Mass Spectrometry

The tandem mass spectrometry (MS/MS, MS^2) is based on a concatenation of two mass analyzers to obtain more accurate results. Let's assume that a mass analyzer is replaced by a chain *mass analyzer 1*, *collision chamber* and *mass analyzer 2* (Fig. 2.5).

The first analyzer separates peptide ions by $\frac{m}{z}$ values. In the collision chamber, peptide ions collide with molecules of an inert gas (e.g., argon or xenon) and the ions are fragmented into peptide fragment ions. The second mass analyzer separates peptide fragment ions by $\frac{m}{z}$ values. Finally, a mass spectrum of peptide fragment ions is generated for each peptide ion.

In principle, any analyzers described in Sec. 2.3.2 can be used while a tandem mass spectrometer can be constructed from different types of analyzers. We can meet with analyzers TOF-TOF, Q-TOF (Quadrupole TOF), Q-Trap (Quadrupole Ion Trap) or QQQ (Quadrupole-Quadrupole-Quadrupole; the second quadrupole plays the role of a collision chamber), etc. Generally, combinations of analyzers impact the accuracy of the machine and its price. Moreover, different instruments are more or less suitable for different experimental setups [5]. Even though it is not very common, more than two mass analyzers can be concatenated to generate MS^n spectra where n is the number of concatenated analyzers.

2.4.1 Tandem Mass Spectrum

In shotgun proteomics (Sec. 3.3.1), peptides are subjected to a mass spectrometer after a chromatographic separation. The peptides are charged and separated by their $\frac{m}{z}$ ratios. Intensities of peptides are detected and thus a *mass spectrum* is obtained. The mass spectrum is a list of peaks where each peak is represented by a pair ($\frac{m}{z}$ ratio, intensity) corresponding to a peptide ion.

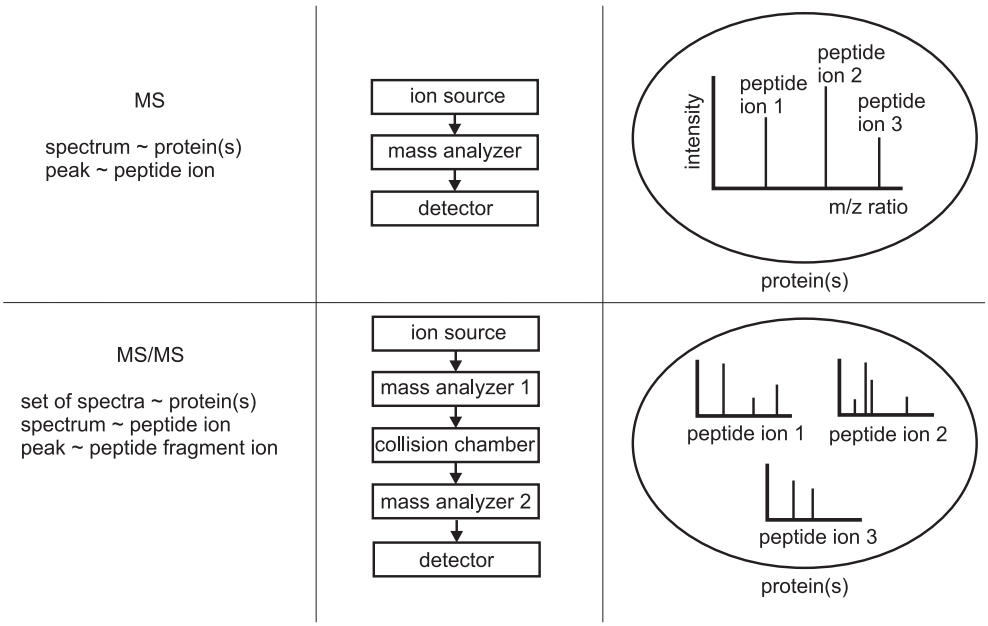


Figure 2.5: Differences between MS and MS/MS

For a small mixture of purified proteins, one spectrum can be generated where each peak corresponds to a peptide ion. This kind of mass spectrometry is known as the *single* mass spectrometry (MS). The identification of protein sequences is commonly realized by a comparison of an experimentally taken spectrum with a library of known spectra. The method is known as the *peptide mass fingerprinting* [18].

In case of MS/MS, a set of spectra is generated where each spectrum corresponds to a peptide ion (Fig. 2.5). In contrast to MS, each peak is represented by the pair ($\frac{m}{z}$ ratio, intensity) corresponding to a peptide fragment ion. The mass corresponding to the $\frac{m}{z}$ value of a peptide ion being split into fragments is called the *peptide precursor mass*. The tandem mass spectrometer estimates the *charge* of the precursor ion from $\frac{m}{z}$ values of fragment ions. For each precursor ion, the *retention time* is also determined.

In a collision chamber of a tandem mass spectrometer, a peptide ion is commonly split into fragments at a peptide bond (Fig. 2.7). We distinguish between *N-terminal* fragment ions containing the terminal group NH_2 and *C-terminal* ions containing the terminal group $COOH$. Since peptide ions are split at different peptide bonds along the *backbone*, the tandem mass spectrum contains

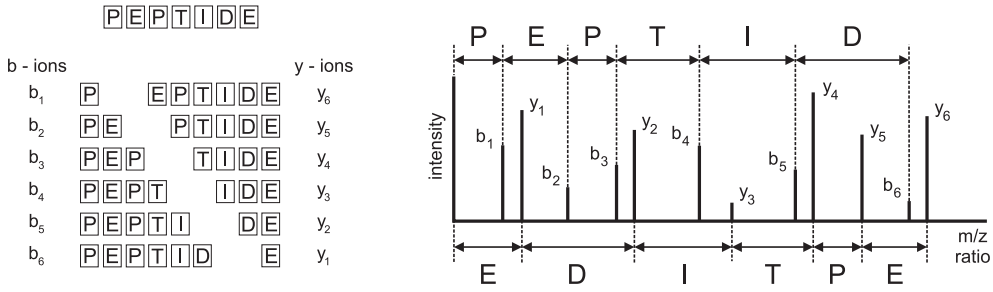


Figure 2.6: An example of a tandem mass spectrum

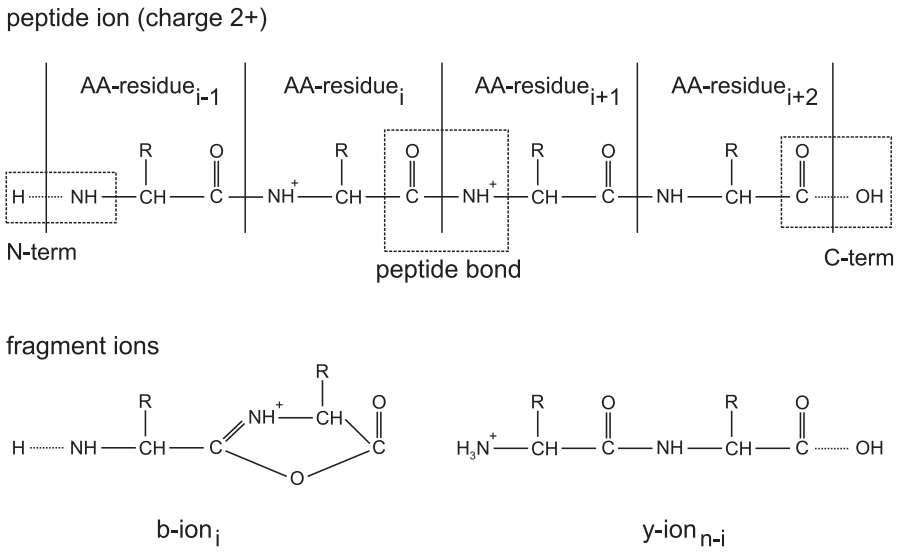


Figure 2.7: Splitting of a peptide ion into fragment ions (n is the number of AA-residues in the peptide)

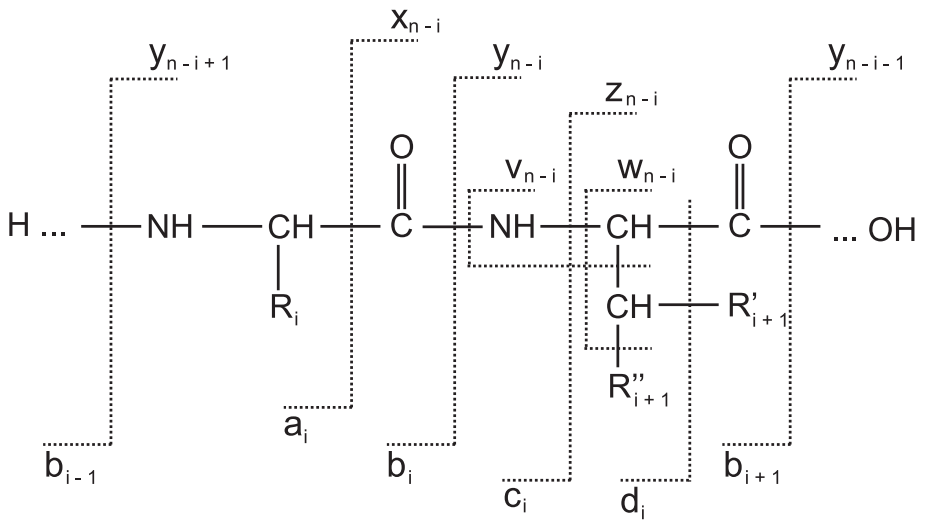


Figure 2.8: Other types of fragment ions

complementary fragment ion series with well predictable structure. The most known types of fragment ions are *b-ions* (N-terminal) and *y-ions* (C-terminal) which are commonly the most important for correct identification of a peptide sequence corresponding to the spectrum (Fig. 2.6). The fragment ions form series where the difference between any two neighboring ions in a series corresponds to a mass of an amino acid residue (AA-residue). However, a peptide ion does not have to be split exactly at a peptide bond (Fig. 2.8). In these cases, other types of fragment ions can arise like a-ions/c-ions (N-terminal), or x-ions/z-ions (C-terminal). An overview of fragment ions is proposed in Tab. 2.2 [11].

Note that a mass spectrometer detects only charged ions. When the charge of a peptide ion is at least 2^+, it is likely that both types of fragment ions (e.g., b-ion and y-ion) arise from a single peptide ion (Fig. 2.7). This is advantageous for identification of peptide sequences because of higher completeness of fragment ion series, and it happens when ESI is used as the ionization technique (Sec. 2.3.1).

The identification of peptide sequences from spectra is often complicated because many fragment ions with unpredictable structure may arise. These frag-

Fragment ion type	**Composition**	$\frac{m}{z}$ **value**	**Frequency**
a	$\Sigma + H - CO$	$M_r(\Sigma) - 27.00216$	quite common
b	$\Sigma + H$	$M_r(\Sigma) + 1.00794$	common
c	$\Sigma + H + NH + H + H$	$M_r(\Sigma) + 18.0385$	rare
x	$\Sigma + OH + CO$	$M_r(\Sigma) + 45.01744$	rare
y	$\Sigma + OH + H + H$	$M_r(\Sigma) + 19.02322$	very common
z	$\Sigma + OH - NH$	$M_r(\Sigma) + 1.99266$	very rare
doubly charged ion	ion + H	($\frac{m}{z}$ of ion + 1.00794)/2	very common
triply charged ion	ion + $H + H$	($\frac{m}{z}$ of ion + 2.01588)/3	rare
a^*, b^*, y^*	ion − NH_3	$\frac{m}{z}$ of ion − 17.03056	w.r.t. ion type
a^o, b^o, y^o	ion − H_2O	$\frac{m}{z}$ of ion − 18.01528	w.r.t. ion type
d_i	a_{i+1} − part of the sideways chain	-	-
v	y − sideways chain	-	-
w	z − part of the sideways chain	-	-

Table 2.2: Fragment ions compositions and $\frac{m}{z}$ values (Σ is the sum of amino acid residues; $M_r(\Sigma)$ is the relative molecular mass of amino acid residues)

ment ions are regarded as a noise and they form up to 80% of all peaks in a spectrum. The intensity may help to differentiate between more and less significant peaks in a spectrum. However, it is not fully guaranteed that a peak with low intensity must be a noise one, and the peak with high intensity must be a signal peak. Another problem is the incompleteness of the y-ion and b-ion series which causes a loss of information about the order of amino acids.

2.4.2 Modifications in Tandem Mass Spectra

The identification of peptides is often complicated due to modifications in the mass spectra which change masses of amino acids and thus cause shifts of $\frac{m}{z}$ values (Fig. 6.5) [19]. Modifications can be artificially added to an "in vitro" sample because they enable more precise analysis (e.g., carbamidomethylation of cysteine). They can arise during the sample preparation or during the mass analysis. A special group of modifications are post-translational modifications (PTMs) which arise during the lifetime of a protein molecule and they give new properties to proteins, make stable conformations of proteins, regulate protein functions, etc. We distinguish two kinds of modifications – *fixed* and *variable*. Fixed modifications change all amino acids of the same type, e.g., carbamidomethylation of cysteine. Variable modifications do not have to change all amino acids of the same type, e.g., oxidation of methionine.

The database UNIMOD gathers discovered protein modifications for the mass spectrometry [20]. At the time of writing this thesis, there was about a thousand of known modifications.

Chapter 3

Algorithms for Processing of Mass Spectra

Because of many inaccuracies in mass spectra, the identification of peptides and proteins is a non-trivial task. In this chapter, the methods used for a preprocessing of spectra prior to identification are briefly mentioned. Then the commonly used approaches for identification and quantification of peptides/proteins, and approaches for a statistical evaluation of results produced by different search engines are described. Finally, the existing frameworks for complex (HP)LC-MS/MS data analysis and management are briefly introduced.

3.1 Preprocessing of Spectra

Raw spectra produced by a spectrometer contain many noise peaks and they resemble an analog signal, thus a preprocessing of spectra is commonly applied before peptides and proteins can be identified from the spectra [13]. The *peak picking* is commonly used to recognize *signal peaks* (i.e., peaks corresponding to y-ions and b-ions) in raw data and to form *peak lists* [21] [22]. The peak picking is often done by vendor software bundled with the machine. However, the process is imperfect in practice and preprocessed peak lists commonly contain tens to hundreds of peaks.

Once the peak lists are formed from raw data, the *deisotoping* is commonly used to reduce peaks belonging to the same fragment ions [5]. For example, peaks having the difference of $\frac{m}{z}$ values equals to 1 Da are very likely different isotopic forms of the same fragment ion and can be represented by one peak.

Before the identification of peptide sequences from the mass spectra, it is advantageous to utilize advanced methods eliminating noise peaks from mass spectra and methods eliminating low-quality and redundant spectra from sets of spectra. The *peak selection heuristics* can be used to eliminate noise peaks in the spectra. The *spectrum quality filtering* eliminates the low-quality spectra, and the *clustering of spectra* can be used to remove low-quality spectra and spectra corresponding to the same peptides [23] [24].

3.1.1 Peak Selection Heuristics

Two simple heuristics based on a selection of a specified number of peaks with highest intensities are described below. More sophisticated heuristics for the denoising of mass spectra were proposed, e.g., in [25] [26].

Peaks with Highest Intensities

The heuristic consists in a selection of p peaks which correspond to peaks with highest intensities in a query spectrum. The heuristic is not very good in practice because the highest intensity peaks are not distributed evenly.

Peaks with Highest Intensities in a Window

A more sophisticated heuristic which splits the range of $\frac{m}{z}$ values in a query mass spectrum into windows of specified size w (e.g., $w = 50\,\mathrm{Da}$). The p peaks with highest intensities are selected from each window, e.g., $p = 5$. Finally, m peaks with highest intensities are chosen from all pre-selected peaks to limit the number of peaks in the query spectrum (e.g., $m = 50$).

3.1.2 Spectrum Quality Filtering

The spectrum quality filtering is a way how to remove low-quality spectra from the set of spectra produced by a spectrometer [27] [28] [5]. Reasons why low-quality (i.e., uninterpretable) spectra are presented in the query sets of spectra are different. For example, many y-ions and b-ions in a spectrum are missing, the fragmented precursor ion does not have to be a peptide, the peptide may be modified in a way that is not taken into account by the search engine, or the peptide is missing in the searched database.

The quality filtering analyzes many parameters of spectra (the number of peaks, the number of peaks with relative intensity > 0.1, the intensity difference between top two peaks, the precursor mass, the charge of precursor ion, the number of complementary y-ions and b-ions, etc.) and assigns a score to each spectrum. Only spectra exceeding a score threshold are further analyzed while the other spectra are ignored. Since mass spectrometers from different manufacturers use different physical principles, the significance of parameters differs from instrument to instrument. Thus, the score heavily depends on the mass spectrometer which was used to capture the spectra [27].

A machine learning technique (i.e., a *classifier*) is commonly used to make a decision whether a spectrum is a "good" or "bad" quality one. The classifier requires a set of parameters and a training set of spectra for which the qualities are known. Once the classifier is trained, it can be used on a testing set. The commonly used classifiers are, e.g., Bayesian classifiers, support vector machines, neural networks, quadratic discriminant analysis or decision trees. However, the classification is not perfect in practice and thus some low-quality spectra may be classified as "good" and vice versa. An advantage is that proteins commonly contain many peptides and thus wrong classifications of some spectra do not have to significantly impact the identified protein sequences.

3.1.3 Spectrum Clustering

The spectrum clustering is another way how to remove low-quality spectra. Moreover, since a mass spectrometer generates multiple spectra corresponding to a peptide sequence, the clustering can be also used to eliminate the spectra corresponding to the same peptide (i.e., *sibling spectra*) [29] [30] [31] [32] [33]. An advantage is that the clustering is independent on the properties of different instruments because the spectra from different sources are processed the same way, i.e., without the knowledge of significance of particular parameters.

To form the clusters, a pairwise similarity or distance function between the spectra must be defined. Suitable similarity functions are, e.g., the cosine similarity (Sec. 6.1.1), the parameterized Hausdorff distance (Sec. 6.1.3) or the sigmoid similarity (Eq. 3.3) [29]. In Eq. 3.1, the variables l and p impact the location and the pivot of the sigmoid curve. In Eq. 3.2 and Eq. 3.3, $a_i \in A$ and $b_j \in B$ are peaks of spectra A and B, $dist(a_i, b_j)$ computes the absolute $\frac{m}{z}$ difference of peaks, $I(a_i)$ is the intensity of the peak a_i, $f(\cdot,\cdot)$ and $g(\cdot,\cdot)$ can be substituted with the minimum, maximum or average.

$$sigm(x) = \frac{1}{1 + e^{\frac{x-l}{p}}} \tag{3.1}$$

$$s(A,B) = \sum_{i=1}^{m}\sum_{j=1}^{n} sigm(dist(a_i, b_j))g(I(a_i), I(b_j)) \tag{3.2}$$

$$S(A,B) = \frac{s(A,B)}{f(s(A,A), s(B,B))} \tag{3.3}$$

Since the spectra corresponding to a peptide sequence are similar, they form a cluster. When a cluster contains only one spectrum, it is called the *singleton*. Spectra in singletons can be regarded as noise. However, a set of spectra from a MS/MS run contains also many spectra which form singletons but they can be assigned to peptides. A disadvantage in these cases is that the clustering may cause a loss of some interpretable spectra.

One of the best-known clustering algorithms is the K-means algorithm [34]. This algorithm is not very suitable for clustering of mass spectra because we cannot predict the number of clusters K before the clustering [30]. Moreover, its time complexity is $O(NKd)$, where N is the number of spectra in the query set and d is the dimensionality. The K-means is not suitable for large query sets and high-dimensional data what is exactly the case of mass spectra (usually containing many peaks/dimensions).

A better clustering algorithm for mass spectra is the hierarchical clustering [30] [34]. The hierarchical clustering can be based on *complete linkage* where all spectra in a cluster are pairwise similar. When an object in a cluster is similar to at least one other object, the clustering is called *single linkage*. A disadvantage of single linkage is that "chains" of similar objects may arise among the clusters. On the other hand, the single linkage is less space-consuming because not all pairwise distances need to be accessible simultaneously. An intermediate criterion can be also employed when an object in a cluster must be similar to at least k other objects. A disadvantage of the hierarchical clustering on large query sets of spectra is the time complexity $O(N^2)$. An example of a bit more efficient

algorithm is the density clustering (DENCLUE) [35] with the time complexity $O(N \log N)$, which is capable of tackling high-dimensional data (what is exactly the case of mass spectra) and which is robust when dealing with noise data.

Once the spectra are clustered, it is suitable to let one spectrum represent a cluster. In general, two ways can be used to obtain a *representative spectrum* of a cluster. First, the best (e.g., the most intense) spectrum in the cluster can be selected. Second, a representative spectrum can be obtained by aggregating all spectra in a cluster. In practice, none of these methods has significant advantages over the other method.

3.2 Identification of Peptides

The annotation of query mass spectra with peptide sequences is often realized by means of a similarity search in databases of theoretical spectra generated from databases of known protein sequences [36], by means of a similarity search in libraries of annotated experimental spectra [37] [38] [39] [40] [41] or by *de novo* peptide sequencing (Sec. 3.2.2).

3.2.1 Similarity Search

When the similarity search in a database of theoretical spectra is employed, a query spectrum is compared with all theoretical spectra (i.e., a sequential scan of the database is performed) using a similarity function [42] [43] [44]. The theoretical spectrum with the best score is selected to form a PSM (peptide-spectrum match). In further sections, we describe the similarities used in state-of-the-art tools SEQUEST, MASCOT MS/MS Ions Search, OMSSA and X!Tandem. Apart from these tools, there are also MyriMatch [45], ProteinProspector MS-Tag [46], Morpheus [47] (designed for high-resolution data), and a vast number of other tools [48] [36].

Commonly, the user must select PTMs which will be supported during the similarity search in a database. In this case, theoretical spectra of modified peptides are generated and compared with query spectra. However, there are also approaches supporting the blind search of spectra with modifications like spectral convolution or spectral alignment [49] [19], InsPecT [50] and others [51] [52] [53]. A novel approach is based on a detection of possible modifications in a query set of spectra from differences of $\frac{m}{z}$ values of precursor ions in a query set [54].

SEQUEST

SEQUEST [55] [5] employs two types of scores – the preliminary score S_p and the cross-correlation score X_{corr}. Since X_{corr} is computationally more time consuming than S_p, S_p is used to filter out the theoretical spectra which cannot match a query spectrum and then X_{corr} is computed for remaining theoretical spectra. The preliminary score S_p is defined by Eq. 3.4 where n_i is the number of $\frac{m}{z}$ values in a query spectrum which are paired with $\frac{m}{z}$ values in a theoretical spectrum, $\sum_{m=0}^{n_i} i_m$ is the sum of their intensities and n_t is the total number of predicted $\frac{m}{z}$ values in the theoretical spectrum. The division by n_t prevents S_p from an excessive increase for long peptide sequences from which some theoretical spectra

are generated. The continuity of a matched ion series is taken into consideration by β. The initial value $\beta = 0$ is increased by 0.075 for each matched consecutive fragment ions. $\rho = 0$ is increased by 0.15 when an immonium ion of the amino acids histidine, tyrosine, tryptophan, methionine and phenylalanine occurs in the query spectrum.

$$S_p = \left(\sum_{m=0}^{n_i} i_m \right) \left(\frac{n_i}{n_t} \right) (1 + \beta) (1 + \rho), \tag{3.4}$$

The cross-correlation score X_{corr} (Eq. 3.6) is computed using the correlation function $Corr(t)$ (Eq. 3.5) what is the dot product of vectors $\vec{x}$ and $\vec{y}$ where $\vec{y}$ is shifted by t (by default, $t = 75$). $\vec{x}$ corresponds to a query spectrum and $\vec{y}$ to a theoretical spectrum. The vectors contain $\frac{m}{z}$ values rounded to integers. The function avg computes an average of the values in the interval $\langle Corr(\vec{x}, \vec{y}, -t), Corr(\vec{x}, \vec{y}, t) \rangle$.

$$Corr(\vec{x}, \vec{y}, t) = \sum_{i=1}^{n} \vec{x}_i \vec{y}_{i+t} \tag{3.5}$$

$$X_{corr}(\vec{x}, \vec{y}) = Corr(\vec{x}, \vec{y}, 0) - avg(\langle Corr(\vec{x}, \vec{y}, -t), Corr(\vec{x}, \vec{y}, t) \rangle) \tag{3.6}$$

The PSMs with highest S_p and X_{corr} are preferred in the output. However, SEQUEST uses following additional values to score PSMs.

- $\Delta C_n = \frac{X_{cross1} - X_{cross2}}{X_{cross1}}$ where X_{cross1} and X_{cross2} are the first and the second highest correlation values.
- RS_p is the rank got in the preliminary scoring by S_p.
- $Ions$ is the number of matched $\frac{m}{z}$ values divided by the number of $\frac{m}{z}$ values in the theoretical spectrum.
- dM is the difference between the precursor mass of the query spectrum and precursor mass of the theoretical spectrum.

Several approaches exist which combine these different components into a single score [56] [57]. There are also re-implementations of SEQUEST like Crux [58] and others [59].

MASCOT

MASCOT [60] is a popular commercial software for identification of peptide sequences. The details of its probability scoring algorithm for MS/MS spectra were not published. However, the algorithm for MS/MS spectra is based on the algorithm MOWSE (MOlecular Weight SEarch) for single MS spectra. MOWSE is based on a frequency factor matrix F where each row represents an interval of 100 Da of peptide mass and each column represents an interval of 10 kDa of protein mass. A reason for the construction of F is that peptides with low masses occur more frequently than peptides with high masses. Moreover, the frequencies of occurrence depend on the lengths of protein sequences from which the peptides originate.

To determine the frequency factors, the protein sequence database is traversed while the "in silico" digestion of protein sequences into peptide sequences

is performed. Let F be initialized with zeros. When a peptide sequence is being generated, the corresponding item $f_{i,j}$ in F is incremented. When all peptides have been generated, items in each column are transformed to the probability of their occurrence $f'_{i,j}$ (Eq. 3.7).

$$f'_{i,j} = \frac{f_{i,j}}{\sum\limits_{i} f_{i,j}} \tag{3.7}$$

Then the items in a column are normalized by the maximum value in the column (Eq. 3.8). By the normalization, we get the items $m_{i,j}$ of a new matrix M (MOWSE factor matrix).

$$m_{i,j} = \frac{f'_{i,j}}{\max\limits_{i} f'_{i,j}} \tag{3.8}$$

Finally, the score of a protein is determined by Eq. 3.9 where M_{prot} is the mass of a protein, n number of peaks in the spectrum which correspond to peptides included in the protein and the normalization value 50 kDa is used to protect the score from a large increase for long protein sequences.

$$score = \frac{50,000}{M_{prot} \times \prod\limits_{n} m_{i,j}} \tag{3.9}$$

OMSSA

The Open Mass Spectrometry Search Algorithm (OMSSA) calculates the E-value as a scoring function of PSMs [61]. The calculation of E-value is based on characteristics of random matches of $\frac{m}{z}$ values between a query and a theoretical spectrum. The distribution of random matches allows to determine the significance of a PSM as the probability that the PSM is random. A low probability implies that the PSM is a significant hit. The distribution of matches of $\frac{m}{z}$ values is fit by the Poisson distribution and is calculated separately for different charge states of fragment ions.

Let's assume the charge state 1^+. Let o be the smallest measured $\frac{m}{z}$ value and r be the highest measured $\frac{m}{z}$ value. When t is the $\frac{m}{z}$ error tolerance, the maximum possible number of matches of $\frac{m}{z}$ values is $\frac{r-o}{2t}$. If m is the neutral mass of the precursor, we are trying to match $h\frac{(r-o)}{m}$ theoretical $\frac{m}{z}$ values to v experimental where t is the total number of theoretical $\frac{m}{z}$ values. A mean is then calculated as shown in Eq. 3.10 for the Poisson distribution (Eq. 3.11) where x is the number of matched $\frac{m}{z}$ values between a query and a theoretical spectrum.

$$\mu_1 = \left(\frac{2t}{r-o}\right)\left(\frac{h(r-o)}{m}\right)v = \frac{2thv}{m} \tag{3.10}$$

$$P(x,\mu) = \frac{\mu^x}{x!}e^{-\mu} \tag{3.11}$$

When singly and doubly charged fragment ions (i.e., 1^+ and 2^+) occur in query spectra and are generated into the theoretical spectra, then two separate ranges of $\frac{m}{z}$ values are used to calculate the mean. The $\frac{m}{z}$ range A above $\frac{m}{2}$ which

contains only charge 1^+ fragment ions and the range B below $\frac{m}{2}$ which contains fragment ions with both charges 1^+ and 2^+.

In the range A, the number of possible matches is $\frac{r-\frac{m}{2}}{2t}$ and we are trying to match $h\frac{r-\frac{m}{2}}{m}$ theoretical $\frac{m}{z}$ values to $v\frac{r-\frac{m}{2}}{r-o}$ experimental $\frac{m}{z}$ values. The mean μ_A is calculated by Eq. 3.12.

$$\mu_A = \left(\frac{2t}{r-\frac{m}{2}}\right)\left(\frac{h(r-\frac{m}{2})}{m}\right)\left(\frac{v(r-\frac{m}{2})}{r-o}\right) = \frac{2thv}{m}\frac{r-\frac{m}{2}}{r-o} \tag{3.12}$$

In the range B, the number of possible matches is $\frac{\frac{m}{2}-o}{2t}$ and we are trying to match $h\frac{\frac{m}{2}-o}{m}$ singly charged fragment ions and $h\frac{\frac{m}{2}-o}{\frac{m}{2}}$ doubly charged fragment ions to $v\frac{\frac{m}{2}-o}{r-o}$ experimental $\frac{m}{z}$ values. The mean μ_B is calculated by Eq. 3.13.

$$\mu_B = \left(\frac{2t}{\frac{m}{2}-o}\right)\left(\frac{h(\frac{m}{2}-o)}{m} + \frac{h(\frac{m}{2}-o)}{\frac{m}{2}}\right)\left(\frac{v(\frac{m}{2}-o)}{r-o}\right) = \frac{6thv}{m}\frac{\frac{m}{2}-o}{r-o} \tag{3.13}$$

The resulting mean μ_2 for charge states 1^+ and 2^+ is defined by Eq. 3.14.

$$\mu_2 = \mu_A + \mu_B = \frac{2thv}{m}\frac{r+m-3o}{r-o} = \mu_1\frac{r+m-3o}{r-o} \tag{3.14}$$

The OMSSA further increases sensitivity and efficiency using the following idea – at least one $\frac{m}{z}$ value in a theoretical spectrum must match one of the n peaks with highest intensity in a query spectrum ($n = 3$, by default). This idea changes the probability distribution. Let $q = \frac{n}{v}$ be the probability of a match of $\frac{m}{z}$ values between a theoretical and query mass spectrum, then the probability distribution P' is defined by Eq. 3.15 where the normalization factor Q is defined by Eq. 3.16.

$$P'(x,\mu) = \frac{1}{Q}\left(1-(1-q)^x\right)P(x,\mu) \tag{3.15}$$

$$Q = \sum_x \left(1-(1-q)^x\right)P(x,\mu) \tag{3.16}$$

The probability that a PSM is not random is defined by Eq. 3.17, where y is the number of matches of $\frac{m}{z}$ values between the theoretical and query mass spectrum, and $z = 1$ or $z = 2$ depending on the fragment ion series employed.

$$\sum_{x=0}^{y-1} P'(x,\mu_z) \tag{3.17}$$

When the query spectrum is compared against N theoretical spectra, the probability that a PSM is random is defined by Eq. 3.18.

$$1-\left(\sum_{x=0}^{y-1} P'(x,\mu_z)\right)^N \tag{3.18}$$

Finally, the E-value is calculated using Eq. 3.19. For example, E-value equals to 1.0 means that one hit with a score equal to or better than the hit being scored would be expected at random when a query spectrum is compared against N theoretical spectra.

$$E(y,\mu) = N\left(1-\left(\sum_{x=0}^{y-1} P'(x,\mu_z)\right)^N\right) \tag{3.19}$$

X!Tandem

The X!Tandem [62] implements the hyperscore HS as a similarity between a query and theoretical mass spectrum (Eq. 3.20). I_i is the intensity of a peak in a query spectrum and $P_i \in \{0, 1\}$ says whether the peak is predicted in a theoretical spectrum or not. The hypergeometric distribution of matched $\frac{m}{z}$ values is assumed and thus factorials of the number of matched b-ions N_b and the number of matched y-ions N_y are used.

$$HS = \left(\sum_{i=0}^{n} I_i P_i \right) N_b! N_y! \tag{3.20}$$

3.2.2 De Novo Peptide Sequencing

Query mass spectra can be interpreted also using graph algorithms (without any reference database). Such approaches are called *de novo* peptide sequencing [63] [64] and they are based primarily on the detection of y-ions and b-ions series. A graph is constructed over a query spectrum where a node corresponds to a peak (its $\frac{m}{z}$ value) and an edge is ranked with the $\frac{m}{z}$ difference between two $\frac{m}{z}$ values corresponding to connected nodes. The paths in the graph with most edges are selected whose weights best fit the masses of amino acids. In Fig. 3.1a, edges corresponding to amino acids and pairs of amino acids are shown for a peak with $\frac{m}{z}$ value equals to 260. The complete graph created over the query spectrum is shown in Fig. 3.1b.

A drawback is that many paths and thus many peptide sequences can be assigned to a query spectrum and the number of identified peptide sequences can be low. This is due to noise peaks, modifications of amino acids, substitutions of amino acids with equal or similar masses (also substitutions of pairs or triplets of amino acids), and the fact that some of y-ions or b-ions may never arise. Examples of amino acids and pairs of amino acids with equal or similar masses are shown in Tab. 3.1. Note that masses are equal when the numbers and types of atoms occurring in the amino acids are equal.

The completeness of y-ions or b-ions series impacts the number of identified peptides because the difference between two neighboring peaks in one series corresponds to the mass of an amino acid. For example, when peaks y_3 and b_4 are

Amino acid(s)	**Residue mass [Da]**		**Amino Acid(s)**	**Residue mass [Da]**
L	113.08407	⇔	I	113.08407
Q	128.05858	↔	K	128.09497
A + G	128.05858	⇔	Q	128.05858
A + G	128.05858	↔	K	128.09497
G + G	114.04294	↔	N	114.04293
G + V	156.08989	↔	R	156.10112
A + D	186.06407	⇔	E + G	186.06407
A + D	186.06407	↔	W	186.07932
E + G	186.06407	↔	W	186.07932
S + V	186.10045	↔	W	186.07932
S + S	174.06406	↔	C (+5H+3C+O+N)	174.04629
F	147.06842	↔	M (+O)	147.03539

Table 3.1: Amino acids and pairs of amino acids with equal/similar masses (equal masses are denoted by ⇔ and similar masses by ↔)

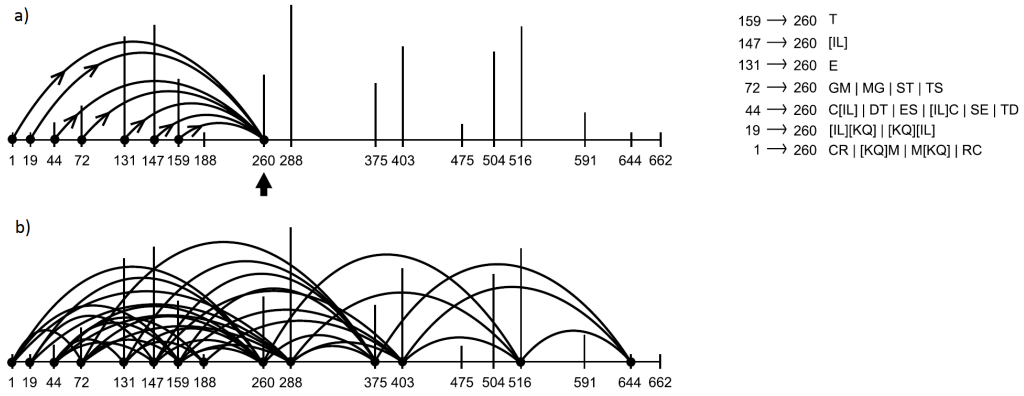


Figure 3.1: De novo peptide sequencing

missing in Fig. 2.6, we lose the information about the order of the letters T and I. The letters can be determined from the difference of $\frac{m}{z}$ values between peaks y_2 and y_4 (or b_3 and b_5) but more candidate pairs of amino acids having similar aggregate $\frac{m}{z}$ values can be selected from 20^2 possible pairs of amino acids.

Tools based on de novo peptide sequencing are, e.g., PEAKS [65], PepNovo [66] and Lutefisk [67]. Instead of the de novo, a novel idea is to use hybrid approaches combined with the database search. Examples of these approaches are sequence tag methods (Sec. 4.2) or lookup peaks [68].

3.2.3 Statistical Evaluation

The output of an engine for identification of peptides is a list of PSMs. Even though there is a score for each PSM, a common problem is a substantial overlap between scores between correct and incorrect peptide sequence identifications. A solution consists in the statistical evaluation of PSMs [69].

The most common significance measure in statistics is the *p-value*. Let a *null hypothesis* be that a PSM is incorrect. For example, a p-value of 0.01 means that there is a 1% chance that the null hypothesis is correct, i.e., that the PSM is incorrect. To determine the p-value of a PSM, we have to know which PSMs are correct and which are incorrect. A widely accepted technique is to apply a *target-decoy* approach [69] [70]. The protein sequences in the database are marked as *target*. The *decoy* sequences are generated by reversing or shuffling of target sequences. Another way is to generate random sequences using a Markov model with parameters derived from target sequences. Finally, the decoy sequences are appended to target sequences.

The query spectra are searched against the database of target and decoy sequences. In an ideal case, there should not be an overlap between target and decoy peptide sequences. The p-value is the percentage of decoy peptide sequences which receive a score x or lower (we assume a distance as a scoring function thus the lower score is better). For example, when the score below a threshold $t \leq 0.3$ is assigned to 30 decoy PSMs and 6000 target PSMs, the p-value is 0.005.

A disadvantage of the p-value is a lack of the *multiple testing correction*. Let's assume a query set containing 30,000 spectra. Ideally, the list generated by an engine should contain 30,000 PSMs. However, for the p-value less or equals to

0.005, 0.005× 30,000 = 150 PSMs are obtained at random. A widely accepted method for multiple testing correction is the *false discovery rate FDR*. For a given score threshold t, the *FDR* is a ratio of the number of decoy PSMs to the number of all PSMs having score equals to or better than t (Eq. 3.21).

$$FDR = \frac{\#decoy}{\#decoy + \#target} \tag{3.21}$$

However, an alternative calculation of *FDR* was suggested in [69] where *FDR* is calculated as the ratio of the number of decoy PSMs to the number of target PSMs (Eq. 3.22).

$$FDR = \frac{\#decoy}{\#target} \tag{3.22}$$

Since *FDR* is a property of a set of PSMs, the *q-value* is used as a property of a single PSM. The q-value is defined as the minimum *FDR* threshold at which a given PSM is accepted as correct [69].

Let's assume a pair-wise distance function d. When a query set of spectra is compared against theoretical spectra, a set of PSMs is obtained where d_i is the distance between i^{th} spectrum in the query set and its nearest theoretical spectrum. Let t be a threshold of d and S be a set of PSMs such that $S = \{PSM_i \in S, d_i \leq t\}$. Now assume the following example – when $t = 0.6$ and S contains 5 decoy and 500 target PSMs, the $FDR = 0.01$; when $t = 0.65$ and S contains 5 decoy and 1000 target PSMs, the $FDR = 0.005$. Since the numbers of decoy PSMs are equal in both cases, the q-value is 0.005 for target PSMs.

An alternative to the q-value is the *posterior error probability PEP* what is the probability that a single PSM is incorrect [71]. The difference between *FDR* and *PEP* is shown in Fig. 3.2, where A and B are areas of the distributions for correct and incorrect PSMs. While *FDR* is the ratio of the number of incorrect PSMs with score $\leq t$ (B) to the total number of PSMs with score $\leq t$ ($A + B$), the *PEP* is the ratio of corresponding heights of the distribution, i.e., the number b of incorrect PSMs with score equal to t is divided by the total number of PSMs with score equal to t ($a + b$).

Commonly, statistical or machine learning approaches are used to estimate the *PEP*. The probability model is learned from a set of annotated training data and used to predict *PEP*s of all future test data. When this approach is used, the *PEP* of a PSM having score t is always the same, regardless the query set in which the PSM occurs. On the other hand, the q-value and *FDR* are always dependent on the query set.

The q-value and *PEP* are useful in different scenarios. When proteins expressed in a certain type of cells are investigated (i.e., a group of PSMs is analyzed), the q-value is more suitable measure. When the presence of a specific peptide or protein is analyzed, the *PEP* is more relevant.

3.2.4 Probabilistic Consensus Scoring

It has been shown that different search engines like X!Tandem, OMSSA or MASCOT assign different peptide sequences to query spectra in many cases. Another words, the first sequence in the list of peptide sequences assigned to a query spectrum does not have to correspond to the correct sequence, or the correct

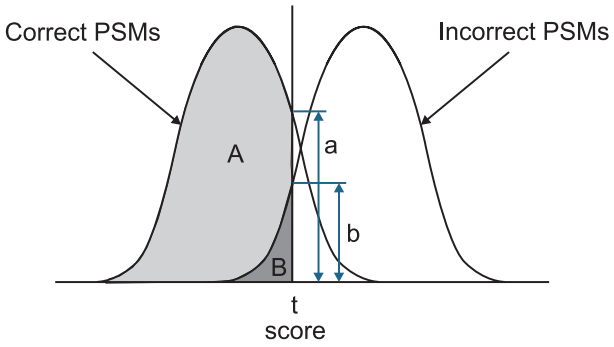


Figure 3.2: Distributions of scores for correct and incorrect PSMs

sequence is missing in the list. The overlap of identified peptide sequences among the engines is usually poor.

When only peptide sequences identified by more than one engine are taken into account, the reliability of identification increases but the sensitivity decreases. When peptide sequences identified by at least one engine are chosen, the sensitivity of identification increases but the reliability is lower. To address this problem, several methods have been developed which combine scores from different engines into one score. Probabilistic consensus scoring is a framework which combines scores from different engines into a joint consensus score [72]. The algorithm works in three steps as follows.

1. The mixture modeling of score distributions is applied to convert scores of peptide sequences from different search engines into probabilities. The distribution of scores is modeled by a two-component mixture model, where a density of incorrectly assigned sequences is modeled as a density of a Gumbel distribution and a density of correctly assigned sequences is modeled as a density of Gaussian distribution.

2. For each peptide sequence p missing in the output of an engine e, the corresponding probability score is estimated. The peptide sequence p' the most similar to p is selected from the output of the engine e. The probability score of p' is assigned to p and weighted by the similarity to p'. The similarity between peptide sequences is computed using global alignment computed by Needleman-Wunsch algorithm [49].

3. A joint consensus score is calculated for each peptide sequence from the probabilities.

When three engines are used, the consensus score is computed as shown in Eq. 3.23. Let s_1 be a probability score of a peptide sequence p matched by engine e_1. Since the most similar peptide sequence to p in the result set of e_1 is p itself, the weight of s_1 is equal to 1. s_2 and s_3 are probabilities of the most similar peptide sequences to p in the result sets of engines e_2 and e_3. The weights α and β are similarities between p and the most similar peptide sequences in the result sets of e_2 and e_3.

$$consensus\ score = \frac{s_1 + \alpha s_2 + \beta s_3}{(1 + \alpha + \beta)^2} \tag{3.23}$$

Other methods based on the combining of scores from multiple engines are, e.g., Scaffold [73], MSblender [74], iProphet [75] or PepArML[76].

3.3 Identification of Proteins

The identification of protein sequences by MS/MS can be performed by two different approaches – by *bottom-up* or *top-down* proteomics [77]. Below, we briefly describe these two approaches.

3.3.1 Bottom-up Proteomics

The bottom-up proteomics (or shotgun proteomics) can be used for identification of small mixtures of purified proteins (up to tens of proteins) as well as for identification of complex protein mixtures (several thousands of proteins) obtained by cell lysis. In the bottom-up proteomics, proteins are enzymatically digested into peptides which are analyzed by LC-MS/MS. The mass spectra are then compared with a database of theoretical peptide spectra generated from a database of protein sequences or analyzed by de novo to determine PSMs. Finally, the PSMs are mapped into protein sequences.

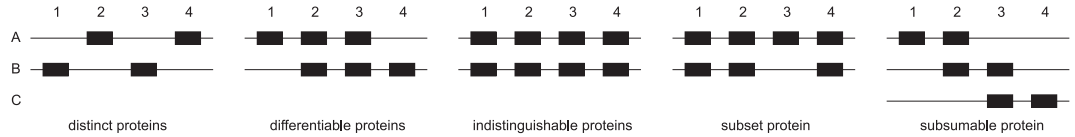


Figure 3.3: Basic scenarios of mapping peptides into proteins

A drawback is that identified peptide sequences commonly do not cover whole protein sequences. Some peptides are too short or too long for detection. However, a major problem are the ionization techniques (Sec. 2.3.1), because peptides must compete for available charges. Properties of peptides and ionization techniques determine whether the peptides are charged or not.

Other problems are caused by non-unique peptides which originate from many proteins. Moreover, the uniqueness of peptides depends on the database of protein sequences from which the peptide sequences are generated, and on the length of peptide sequences because longer sequences become more likely unique.

When protein sequences corresponding to a set of PSMs are being determined, we can look for a *maximal* or for a *minimal explanatory set* of protein sequences [5]. The maximal explanatory set can be easily determined by mapping all PSMs into proteins sequences. Since a protein sequence can be identified by a single PSM, some protein sequences can be identified incorrectly by incorrect PSMs. Moreover, some proteins can be identified even though they are not presented in the analyzed mixture of proteins. The Fig. 3.3 shows possible scenarios when peptide sequences are being mapped into protein sequences. *Distinct* proteins do not share any peptide. *Differentiable* proteins can be distinguished by at least one peptide. *Indistinguishable* proteins share all detected peptides. *Subset* protein contains only peptides which are occurring in another protein. *Subsumable* protein contains only peptides which are occurring in other proteins.

The maximal explanatory set can contain indistinguishable, subset or subsumable proteins even though they are not present in the analyzed mixture [78].

To address this problem, the minimal explanatory set of proteins can be determined using the *maximum parsimony* inference. The idea is to find the smallest set of protein sequences which explain all observed peptides. Thus all indistinguishable, subset and subsumable protein sequences are not included in the explanatory set. However, it may not be necessarily the truth that protein sequences, which are not included in the explanatory set, are not presented in the analyzed mixture. Thus sets of proteins sharing one or more peptides are commonly reported as *protein ambiguity groups*.

Informations about the quantities of peptides (Sec. 3.4) can be also used to prove the presence of a specific protein in the mixture. Let's assume that a protein contains only one peptide which is also included in another protein having assigned more peptides. When the abundance of this peptide is low, the protein match of the former protein is likely random.

Protein Probability Estimates

ProteinProphet is a tool which estimates the probability that a protein sequence is correctly identified or not [79] [36]. The algorithm is based on the maximum parsimony while protein ambiguity groups are reported. To estimate that a protein is correctly identified, *peptide probability estimates* (PPEs) are utilized. PPEs of PSMs are computed by PeptideProphet [56] which converts scores of search engines into probabilities (say, *1-PEP*; Sec. 3.2.3).

The idea of PPEs is to use Bayes' Law to compute the probability $p(+|D)$ to determine whether a PSM is correct or not (Eq. 3.24). For each PSM, the observed data D includes a score or a set of scores generated by some engine(s). $p(D|+)$ and $p(D|-)$ are the probabilities that a PSM is among correctly or incorrectly assigned PSMs, what is determined from the score or the set of scores included in D. The prior probabilities $p(+)$ and $p(-)$ are the overall proportions of correct and incorrect PSMs in the dataset. To compute the probability $p(+|D)$ using Eq. 3.24, a probability distribution of scores generated by the engine(s) must be derived from training data with peptide assignments of known validity or learned from the data itself [56].

$$p(+|D) = \frac{p(D|+)p(+)}{p(D|+)p(+) + p(D|-)p(-)} \tag{3.24}$$

Let's assume that all peptides are unique. The probability P, that a protein identification is correct, can be computed using a sum of probabilities of correct peptide identifications (Eq. 3.25), where $p(+|D_i)$ is a probability of correct peptide identification of a peptide i.

$$P = 1 - \prod_i (1 - p(+|D_i)) \tag{3.25}$$

However, we have to consider that the same peptide sequences can be identified by many spectra. The modification of P is shown in Eq. 3.26, where $p(+|D_i^j)$ is the probability that a peptide identification of a peptide i based on a spectrum j is correct.

$$P = 1 - \prod_i \prod_j (1 - p(+|D_i^j)) \tag{3.26}$$

A problem with Eq. 3.26 is that PSMs are not independent. When a desired peptide sequence is not presented in the database (e.g., because spectra corresponding to a peptide are modified by a PTM), the spectra corresponding to this peptide sequence will likely hit the same but incorrect peptide sequence. A simple solution how to deal with this issue is to include each peptide just once, i.e., from the set of PSMs corresponding to a peptide, only one PSM with the highest probability being correct is used (Eq. 3.27).

$$P = 1 - \prod_i (1 - \max_j p(+|D_i^j)) \tag{3.27}$$

Correct PSMs tend to group into a small set of correct proteins while incorrect PSMs are spread over the entire protein sequence database. ProteinProphet uses the *number of sibling peptides* (NSP) to correct the probabilities of PSMs to address this problem. In this notion, the sibling peptides are those matching the same protein. Peptide identifications with high NSPs are more trustworthy than identifications with low NSPs. NSP_i for the peptide i is the sum of probabilities of correct PSMs of other peptides matching the same protein (Eq. 3.28). m is another distinct peptide matching the same protein and $p(+|D_m)$ is the maximum probability from all PSMs corresponding to the peptide m.

$$NSP_i = \sum_{\{m|m \neq i\}} p(+|D_m) \tag{3.28}$$

Now, we can refine PPEs as shown in Eq. 3.29, where $p(NSP|+)$ and $p(NSP|-)$ are the probabilities of having a particular NSP for correct and incorrect PSMs. $p(+|D)$ and $p(-|D)$ are the uncorrected probabilities for the PSMs being correct and incorrect.

$$p(+|D, NSP) = \frac{p(+|D)p(NSP|+)}{p(+|D)p(NSP|+) + p(-|D)p(NSP|-)} \tag{3.29}$$

$p(NSP|+)$ and $p(NSP|-)$ are computed for the whole dataset (Eq. 3.30), where N is the total number of peptide assignments and $p(+)$ is the prior probability of a peptide identification being correct. NSP values are computed by binning. The probability that a correct peptide assignment has the NSP value in the bin k is computed as a sum over peptides with NSP value in the bin k. $p(+)$ can be computed as a sum over all peptides in the dataset (Eq. 3.31). The NSP distribution for incorrect peptide assignments is computed analogically.

$$p(NSP|+) = \frac{1}{Np(+)} \sum_{\{i|NSP_i \in k\}} p(+|D_i, NSP_i) \tag{3.30}$$

$$p(+) = \frac{1}{N} \sum_i p(+|D_i, NSP_i) \tag{3.31}$$

ProteinProphet considers also *degenerate peptides*, i.e., peptides which can be found in many protein sequences forming protein ambiguity groups. When a

peptide i is included in n different proteins, a relative weight w_i^k of the peptide i in the protein P_k is calculated (Eq. 3.32).

$$w_i^k = \frac{P_k}{\sum_{s=1}^{n} P_s} \tag{3.32}$$

The protein probabilities are computed like in the Eq. 3.27, but the weights w_i^k are taken into account (Eq. 3.33). $p(+|D_i)$ is equal to $\max_j p(+|D_i^j)$.

$$P_k = 1 - \prod_i (1 - w_i^k p(+|D_i)) \tag{3.33}$$

Since Eq. 3.32 and Eq. 3.33 are interdependent, the weights of degenerated peptides are learned iteratively until converge. Initially, weights of peptides are equally apportioned among corresponding proteins.

Similarly, the weights can be taken into account when NSPs are being calculated. The modifications of Eq. 3.28, Eq. 3.30 and Eq. 3.33 for NSPs and weights of degenerated peptides have been proposed in [79]. Again, the equations are computed iteratively until the convergence is obtained.

FDRs of Proteins

A peptide identification engine (Sec. 3.2) returns a set of PSMs. FDRs of PSMs are estimated using a target-decoy approach. When a PSM has the $FDR = 0.01$, it is expected that 1% of all PSMs with the same or better score is incorrect. However, FDRs of proteins are usually higher than FDRs of PSMs because a query set of spectra contains more spectra corresponding to a peptide sequence and because multiple peptide sequences come from a protein sequence (on average, a protein yields 35 peptides of a typical length 10-15 amino acids) [5]. For example, commonly used values for FDRs of PSMs in the range $1\% - 5\%$ are too high because they yield large FDRs of proteins ($> 10\%$) [80]. Another words, there is the 10% chance that an identified protein is incorrect.

In fact, more correct PSMs do not imply better FDRs of proteins. The reason is that the same proteins are being identified with the increasing number of PSMs. Additional PSMs (even though they are correct) do not increase the number of correctly identified proteins but yield hits to random proteins and generate false positives. However, FDRs of proteins can be computed by target-decoy approach like FDRs of PSMs (Sec. 3.2.3) [80] [81].

3.3.2 Top-down Proteomics

In the bottom-up approach, the purpose of digesting proteins into peptides is that peptides are much more suitable for analysis by mass spectrometry. However, many peptides are not ionized and detected what results in a lack of protein sequence coverage by identified peptide sequences. In the top-down proteomics, whole proteins are analyzed without any cleavage into peptides [82] [5]. Thus the approach enables a full protein characterization (the determination of various protein properties, localization of PTMs, etc.), but there are high requirements for

the resolution and accuracy of spectrometers. Moreover, prices and maintenance costs of such machines are high.

The proteins are separated and ionized commonly by ESI (Sec. 2.3.1). Because of ESI, protein ions are usually highly charged (say, up to charge 30^+). Then the protein ions are fragmented into multiply charged b-ions and y-ions. A drawback of this approach is that a *deconvolution* of spectra must be performed. Since there are many peaks corresponding the same b-ion or y-ion having different charges, these peaks must be detected and removed. For example, for a protein having mass 20,000 Da, the peaks having $\frac{m}{z}$ values 667, 691 and 715 can occur for fragment ions having charges 30^+, 29^+ and 28^+ (Eq. 2.1). A drawback is that a reasonable deconvolution is possible only for small mixtures of proteins what limits the high-throughput capabilities of top-down proteomics.

After the deconvolution, the spectra can be analyzed by database search algorithms, de novo or sequence tag methods (Sec. 4.2) [82]. An advantage of the top-down approach is that it yields better characterization of proteins modified by PTMs. Since peaks corresponding to either modified or unmodified fragment ions can be found in a spectrum after its deconvolution, a localization of PTMs in a protein sequence is more straightforward than in the bottom-up approach.

3.4 Quantification of Peptides and Proteins

The quantification of peptides and proteins is an important task in computational proteomics [83] [84] [85] [86]. The quantification is either *relative* or *absolute*. Let's have n samples, each containing a set of components (peptides or proteins). The task is to determine the abundance of components across the samples. When the relative quantification is utilized, the relative changes of intensities of peptide ions are calculated. In case of absolute quantification, a reference additive of known concentration must be available for every analyte to determine the abundance of components from their intensities, thus the absolute quantification is more difficult and less common than relative. The techniques for the relative quantification can be split into *label-based* and *label-free* methods. However, any of these methods can be also adopted for the absolute quantification.

3.4.1 Label-based Quantification

Label-based quantification (LBQ) methods are based on labeling of selected amino acids using stable isotopes of these amino acids. LBQ techniques can be split into *chemical labeling* performed "in vitro" and *metabolic labeling* performed "in vivo". Examples of chemical labeling are ICAT (isotope-coded affinity tags) [87], iTRAQ (isobaric tags for relative and absolute quantification) [88] or labeling by heavy isotopes of oxygen $^{16}O/^{18}O$ [89]. Examples of metabolic labeling are stable isotope labeling by amino acids in cell culture (SILAC) [90] or labeling by heavy isotopes of nitrogen $^{14}N/^{15}N$.

An example of labeling by SILAC is shown in Fig. 3.4 [91]. Two samples are analyzed, where the first sample contains the "light" arginine and the second contains the "heavy" arginine. Another words, cell cultures in the samples are "feeded" by different forms of arginine. The light arginine contains carbons $^{12}C_6$, while its heavy form contains carbons $^{13}C_6$. Since the arginine has six carbons, the

mass difference between its light and heavy form is 6 Da. The samples are mixed before LC-MS/MS analysis. After the analysis, the query set contains spectra of heavy and light peptide ions whose precursor masses and peaks corresponding to fragment ions are shifted by 6 Da. From the differences of intensities of shifted peaks, the abundances of peptides in the samples are calculated. Instead of arginine, the lysine can be used as well. The utilization of these amino acids is advantageous because the trypsin digests proteins into peptides after these two amino acids. In an ideal case, each peptide contains at most one arginine or lysine. The quantification of peptides/proteins from SILAC data is supported, e.g., by MaxQuant [92] and OpenMS [93].

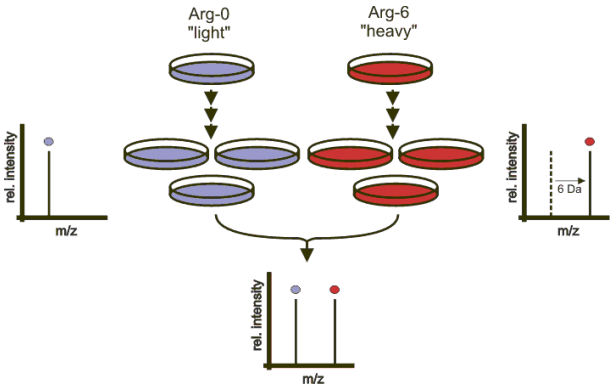


Figure 3.4: Principle of labeling

3.4.2 Label-free Quantification

The label-free quantification (LFQ) does not use any kind of labeling and thus the samples must be analyzed separately, i.e., a LC-MS/MS run is performed for each sample [94] [95]. An overview of tools for label-free quantification is proposed in [96].

The $\frac{m}{z}$ values and retention times of peptide precursor ions obtained in a run form a *map*. An example of a map is shown in Fig. 3.6. Note that not all precursor ions in the map were analyzed by the spectrometer to obtain the fragmentation spectra. Precursor ions within tight intervals of $\frac{m}{z}$ values and retention times in the map commonly correspond to a peptide. Such a two-dimensional region in the map corresponding to a single charge variant of a peptide is called a *feature*. The feature has several attributes – an average $\frac{m}{z}$ ratio, a centroid retention time, an intensity, and a quality value.

The basic principle of LFQ can be summarized as follows:

1. Find features in all maps.
2. Align maps.
3. Link corresponding features.
4. Identify features.
5. Quantify features.

Since more maps are generated from more LC-MS/MS runs, the first step is to determine features in all the maps. In the second step, the maps must be aligned because retention times between any two maps can be shifted. Once the features are determined and maps are aligned, corresponding features across the maps are linked. In the fourth step, the identification of peptides from spectra corresponding to features is performed by peptide identification engines described in Sec. 3.2.

Instead of the fourth step, the identification of peptides can be also performed in the first step – independently and in parallel to the feature finding. This slows down the algorithm because many more spectra must be analyzed by a peptide identification engine. On the other hand, the sensitivity of quantification increases because some peptides can be lost when features are determined without the knowledge of peptides corresponding to the features.

Finally, the peptides/proteins occurring in the samples are quantified. This can be done by computing differences in intensities of features across the maps. The *spectral counting* is another way how to quantify the results, where the spectra corresponding to a peptide/protein are summed [97].

Since the feature finding and maps alignment are crucial parts of LFQ, they are briefly described below.

Feature Detection

The identification of a feature corresponds to the detection of all peaks of precursor ions in the map belonging to a peptide [86] [98]. The main idea is to determine suspicious regions in the map and to fit two-dimensional models to that regions. The algorithm for feature finding can be summarized as follows:

1. **Seeding**. Peaks with highest intensities in the map are selected as "seeds", because they are very likely in features.

2. **Extension**. The peaks around the seeds are conservatively appended to the seeds and create regions. A region grows in all directions simultaneously while the points in its neighborhood with highest priorities are appended. The priority cannot be represented directly by the peak intensity because it must be smaller with increasing distance from the region and missing peaks must be also considered. When a point is appended to a region, a boundary of the region is updated. The extension of the region stops when the priority of neighboring data points is below a certain threshold.

3. **Modeling**. A two-dimensional statistical model is fit to each region to form a peptide feature. The model is based on the Gaussian elution profile [99] in the dimension of the retention time and on the averagine isotope model [100] in the dimension of $\frac{m}{z}$ values.

4. **Adjusting**. The peaks, which do not fit the model of its feature, are removed. A variant of the priority from the extension phase is used for this purpose but the statistical model is also taken into account. The steps 3 and 4 are repeated, until the models of features converge.

Maps Alignment

The optimal alignment between any two maps can be computed by a linear alignment algorithm under the assumption that shifts of retention times between two runs are linear [101]. The alignment of u maps can be performed simply by applying $u-1$ linear alignments onto one reference map.

The reference (or *model*) map is commonly the map with the most features. Let *scene* be a map which is being aligned with the model. The goal is to determine an affine transformation $t(x) = ax+b$, which maps the points $\{s_1, s_2, \ldots, s_k\}$ of a scene onto their nearest neighbors $\{m_1, m_2, \ldots, m_l\}$ in the model. a is the *scale* and b is the *shift* of the mapping. This problem is also known as the *superposition*. The parameters a and b can be recovered from the data using the algorithm called *pose clustering*. The optimal affine transformation t is uniquely determined when each map contains only two points. Then a and b can be computed using the equations $a = \frac{m_1-m_2}{s_1-s_2}$ and $b = \frac{s_1m_2-s_2m_1}{s_1-s_2}$, where $t(s_1) = m_1$, $t(s_2) = m_2$ and $s_1 \neq s_2$.

For large sets of points $\{s_1, s_2, \ldots, s_k\}$ and $\{m_1, m_2, \ldots, m_l\}$, the system of equations is overdetermined and thus approximate values for a and b must be computed. The best approximate solution is such that maps most points from the scene to their nearest neighbors in the model. Another problem is that assignments of matching points between scene and model are not considered by the algorithm. Both problems are solved by a *voting scheme*. Let assume a set of pairs of parameters a and b where each pair corresponds to pairs of objects (s_1, m_1) and (s_2, m_2), we obtain a set T of affine transformations $t_1, t_2, \ldots, t_n$. When the set T is applied on a set of pairs of objects (s'_1, m'_1) and (s'_2, m'_2), a "correct" transformation generates clusters of points while other transformations generate randomly distributed points over the plane (a, b). The centroid of a cluster is then used to estimate the optimal transformation.

The voting scheme can iterate over all possible pairs of objects (s'_1, m'_1) and (s'_2, m'_2). However, the time complexity of the algorithm is $\mathrm{O}(k^2l^2)$ what leads to a very slow solution for typical values of $k, l > 1000$. Fortunately, the typical $\frac{m}{z}$ error tolerance ξ is small $\xi \in \langle 0.5, 2\rangle$ Da, thus ranges of $\frac{m}{z}$ values of compared pairs of objects are tight and the algorithm is efficient.

In practice, shifts of retention times between two maps do not have to be linear, thus an alternative linear alignment and its extension to a non-linear alignment have been proposed. The idea of non-linear alignment is to compute the linear alignment using the pose clustering and then compute more accurate local alignments between small windows of points using local linear regression method (loess) [102].

3.5 Frameworks for Shotgun Proteomics

The processing of data from shotgun proteomics is a non-trivial task which requires many steps like preprocessing of spectra, identification of peptides, statistical evaluation of results, detection of features, quantification of peptides and proteins, etc. To address this problem, a couple of software solutions has been proposed which separate each step into a specialized tool and which allow a creation of complex pipelines from these tools. Since standardized formats of input

and output files of the tools are used, the tools can be easily interconnected with respect to users' requirements.

For example, the *Trans-Proteomic Pipeline* (TPP) is a freely available open-source proteomics data analysis pipeline developed at the Institute for Systems Biology at the Seattle Proteome Center [103]. Another framework is *The OpenMS Proteomics Pipeline* (TOPP) [104] based on OpenMS [93]. OpenMS [93] is an open-source C++ library for LC-MS/MS data management and analyses. Since the application SimTandem proposed in Sec. 8.2 has been designed for TOPP, we briefly describe this framework.

The TOPP contains a set of tools which can be combined into pipelines. The tools cover a wide range of areas, for example, there are tools for file format conversions, generating of decoy databases, preprocessing of spectra, digesting of protein sequences into peptide sequences, retention time prediction, wrapping of the commonly used tools for peptide identification (e.g., MASCOT, OMSSA, X!Tandem), statistical evaluation of PSMs, label-free and label-based quantification and many others. One of the tools is also *The OpenMS Proteomics Pipeline ASsistant* (TOPPAS) what is a GUI for a graphical composition of pipelines of various tools [105]. An example of a simple peptide identification pipeline with statistical evaluation of results is shown in Fig. 8.6.

A more complex pipeline for label-free quantification is shown in Fig. 3.5. Input files in the node 1 are query sets of mass spectra (*.mzML). Input file in the node 4 is the database of protein sequences (*.fasta). The pipeline uses wrappers for three different peptide identification tools *MASCOTAdapterOnline*, *XTandemAdapter* and *OMSSAAdapter*. The lists of PSMs produced by the engines are converted to posterior error probabilities (Sec. 3.2.3). Then *IDMerger* merges lists of PSMs and *ConsensusID* computes consensus scores. *PeptideIndexer* annotates for each search result whether it is a target or a decoy hit, *FalseDiscoveryRate* computes q-values and *IDFilter* selects only those PSMs with q-values less or equal the specified tolerance. In parallel to the identification part of the workflow, *FeatureFinderCentroided* detects features in the input maps. *IDMapper* assigns peptide identifications to the features. When all maps are collected, *MapAlignerPoseClustering* aligns the maps and *FeatureLinkerUnlabeled* groups corresponding features in multiple runs of label-free experiments. Finally, peptide and protein abundances are computed from annotated maps by *ProteinQuantifier*.

Another graphical tool is TOPPView [106], which enables visualizations of 2D and 3D maps of mass spectra query sets, visualizations of fragmentation spectra of peptides, annotations of query sets (*.mzML) with identified sequences (*.idXML), etc. The 2D map of the E. coli query set described in Sec. 7.2.4 is shown in Fig. 3.6, where the axes represent $\frac{m}{z}$ values of peptide precursor ions and their retention times. Each horizontal "smudge" corresponds to a feature/peptide. The 3D map of the same query set is shown in Fig. 3.7, where the third axis represents intensities of peptide precursor ions.

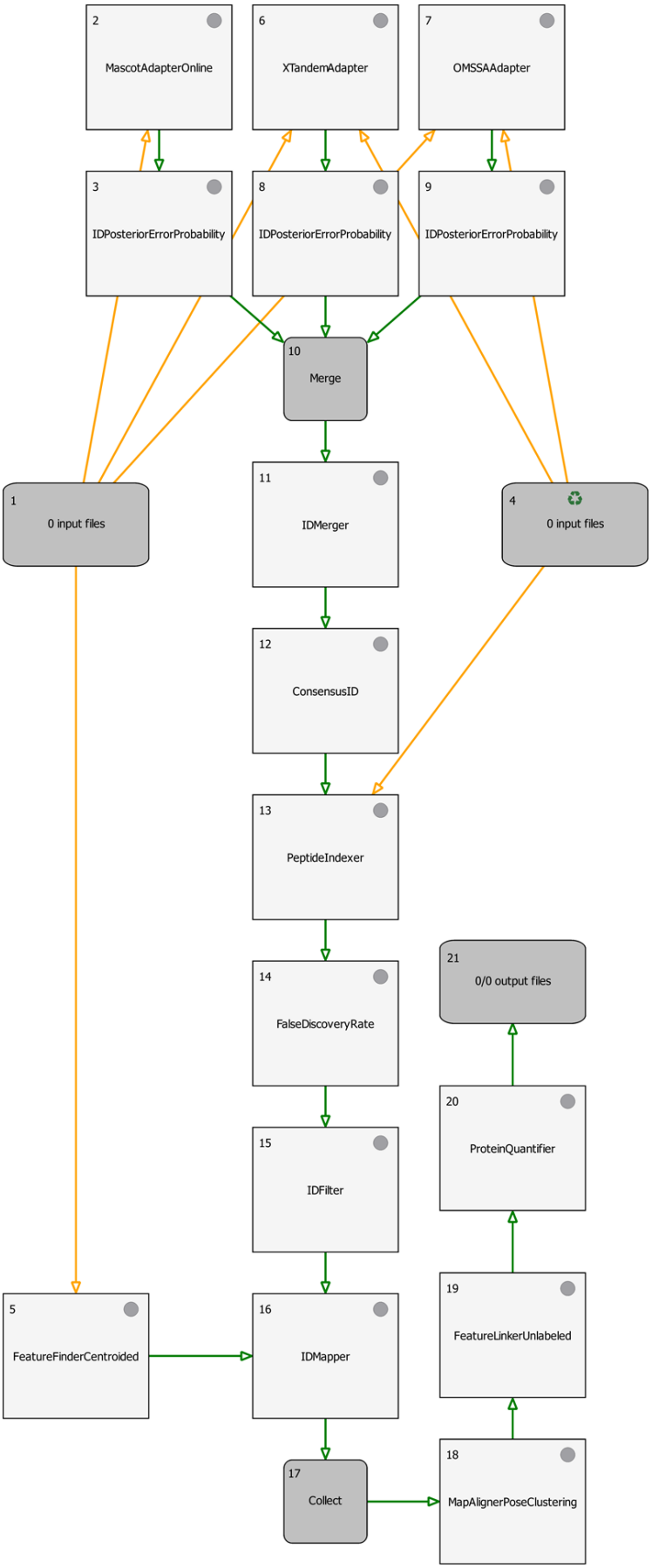


Figure 3.5: An example of a complex pipeline in TOPPAS

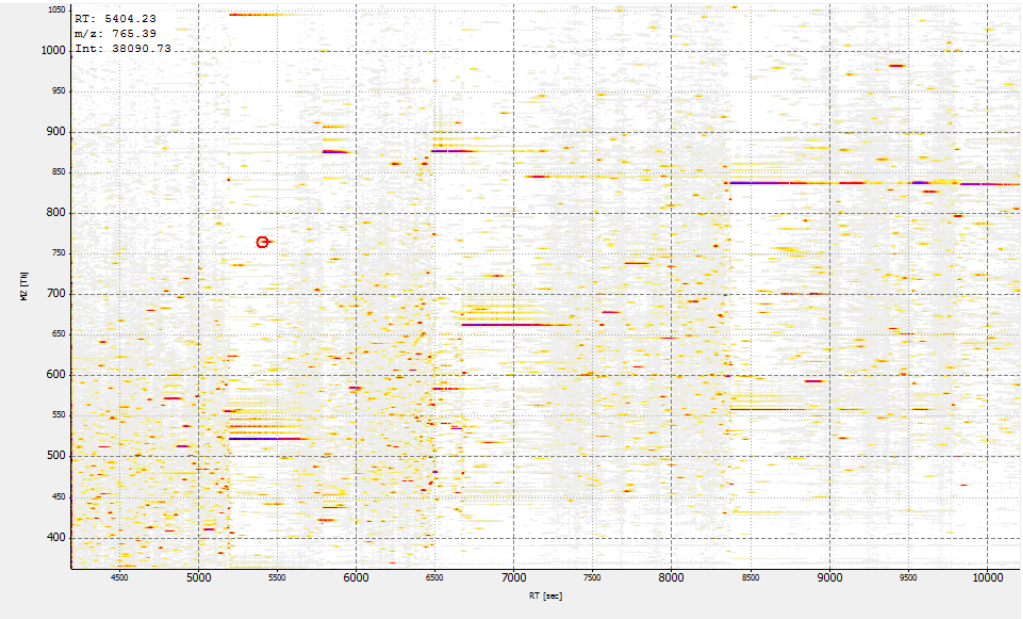


Figure 3.6: 2D visualization of the E. coli query set in TOPPView (horizontal axis – retention time of precursor peptide ions; vertical axis – $\frac{m}{z}$ ratio of precursor peptide ions)

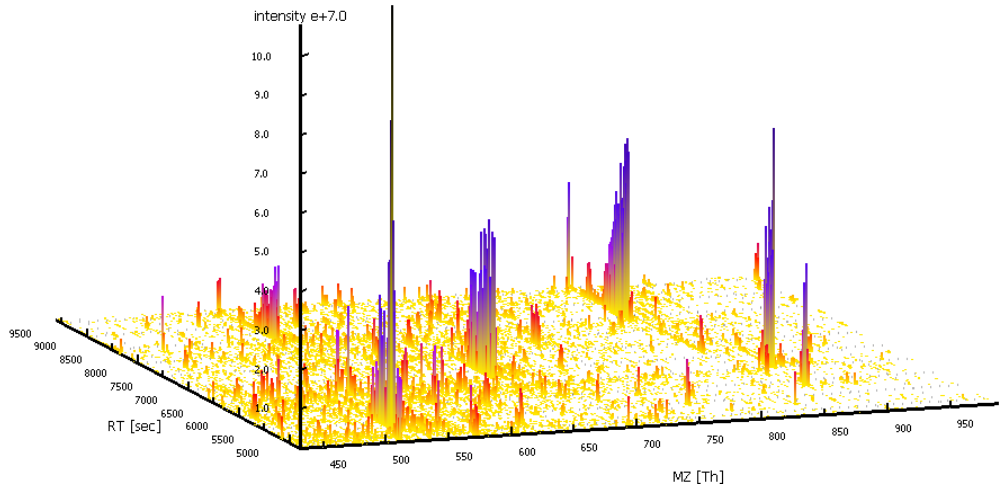


Figure 3.7: 3D visualization of the E. coli query set in TOPPView (in comparison to 2D visualization, the third axis shows the intensity of precursor peptide ions)

Chapter 4

Speeding up the Mass Spectra Database Search

The number of protein sequences in public-available databases grows almost exponentially in recent years (Fig. 4.1). Since large query sets containing tens of thousands of query mass spectra are compared with tens to hundreds of millions of theoretical spectra generated from a database of protein sequences, a sequential scan of the database becomes inefficient. When a query spectrum is compared with n theoretical spectra generated from the database of protein sequences, the time complexity of one-to-all comparisons is linear with the size of the database $O(n)$. When a query set contains m spectra, the time complexity of all-to-all comparisons is $O(nm)$. A solution is to avoid one-to-all comparisons and to find an approach with sub-linear time complexity, i.e., there is a need for utilization of database index structures. In practice, the modeling of an index structure is a non-trivial task due to the noise, modifications and inaccuracies in the mass spectra.

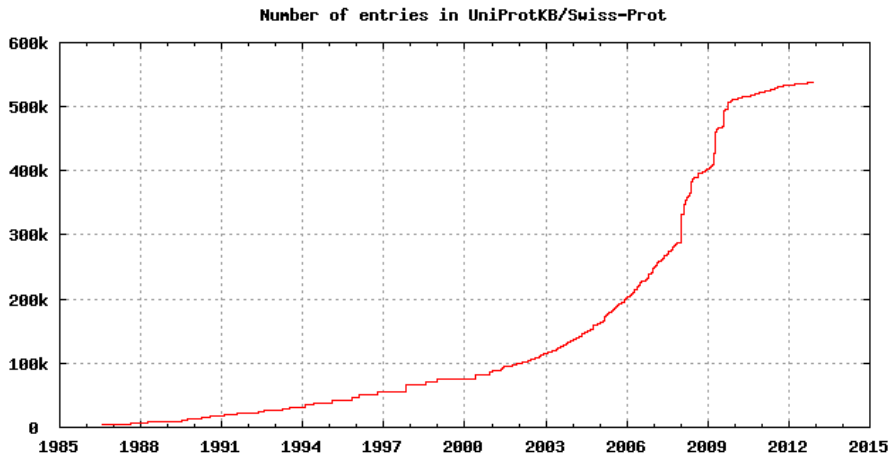


Figure 4.1: Numbers of protein sequences in UniProtKB/Swiss-Prot in years [107]

Several approaches have been proposed which try to avoid one-to-all comparisons. We describe the precursor mass filter, sequence tags, and two approaches based on the properties of metric spaces (Chap. 5) – the tandem cosine distance utilized by MVP-tree and the locality sensitive hashing. We also mention an

approach based on the inverted files and give a brief overview of other speeding up techniques.

4.1 Precursor Mass Filter

A precursor mass filter is the most straightforward method how to speed up the search in a database of theoretical spectra. Since peptide precursor masses are known for both – theoretical and query mass spectra, a query spectrum q does not have to be compared with all theoretical spectra T generated from a database of protein sequences but only with a small subset $T' \subset T$ within a precursor mass error tolerance λ.

λ is a property of an instrument and it is usually given in Da (Daltons) or in ppm (parts per million). Let M_z be an observed $\frac{m}{z}$ value of a precursor peptide ion. The real precursor mass is $M_{real} = M_z \pm \lambda$ when λ is given in Da. The real mass is $M'_{real} = M_z \pm \frac{\lambda}{10^6} M_z$ when λ is given in ppm [5].

Let's assume a theoretical mass spectrum $t \in T'$ having the precursor mass M_t. Let's also assume a query spectrum q having the precursor mass M_q. Only those spectra from T' are compared with q for which $|M_t - M_q| \leq \lambda$. For the efficient determination of T', T is sorted by precursor masses and T' is found by a binary search of the precursor mass of the query spectrum q. Alternatively to the binary search, the precursor masses can be indexed, e.g., by B-tree [108]. When T' is determined, theoretical spectra in T' are compared with the query spectrum q using a mass spectra similarity function such as SEQUEST-like scoring (Sec. 3.2.1), d_{HP} (Sec. 6.1.3), etc. Then the nearest theoretical spectrum to the query spectrum is selected. The query spectrum, the nearest theoretical spectrum and the score determined by the similarity function form a peptide-spectrum match (PSM).

The efficiency of the precursor mass filter depends on the precision of a mass spectrometer. While λ of older instruments is commonly given in Da (i.e., $\lambda = 2\,\text{Da}$) and the number of comparisons of a query spectrum with theoretical spectra can reach tens of thousands to millions of comparisons, λ of modern instruments is given in ppm (i.e., $\lambda = 10\,\text{ppm}$) and the number of comparisons reaches hundreds to tens of thousands of comparisons (Tab. 7.12). Moreover, the precision of modern instruments still increases [109][110][13][14].

4.2 Peptide Sequence Tags

A combination of the similarity search in a database of spectra (Sec. 3.2.1) with de novo peptide sequencing (Sec. 3.2.2) is used in the approach called *peptide sequence tags* [5] [111]. A short peptide subsequence is determined in a query spectrum by de novo peptide sequencing. The subsequence forms so-called peptide sequence tag (PST) which commonly contains 3 amino acids. When the PST is determined, a database of theoretical spectra is searched. However, an additional information about masses, where the tag begins and ends, is necessary to perform the database search. For example, MASCOT Sequence Query [60] uses PSTs in the format "*precursor mass* tag(*start mass*,XXX,*end mass*)". Where XXX is a subsequence of amino acids, the precursor mass is the mass of uncharged

peptide, the start mass is the mass where the subsequence begins, and the end mass is the mass where the subsequence ends. For example, the query spectrum corresponding to the peptide sequence "GHPETLEK" in Fig. 4.2 has the tag of the form "909.99 tag(195.20,PET,522.54)".

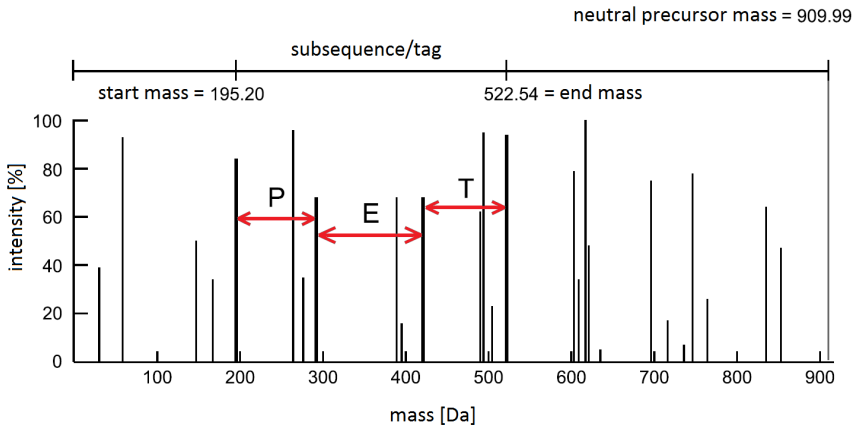


Figure 4.2: Example of a peptide sequence tag in a query spectrum

To speed up the comparisons PSTs with the database of theoretical spectra, an index table can be constructed. The table contains entries for all combinations of amino acids in the sequence tags (e.g., 20^3 entries for tags of the length 3). Each entry has assigned a list of positions of peptide sequences (theoretical spectra) in the database where the entry tag occurs. Drawbacks are that the index must be recomputed when the database is changed, and that the number of PSTs selected from the database grows exponentially with the number of modifications allowed in query spectra. Among others, peptide sequence tags are implemented in PeptideSearch [112], ProteinProspector MS-Seq [46], GutenTag [113] or Inspect [50]. Inspect employs also a *trie* constructed from tags of multiple query spectra to speed up the database search. The trie is a tree data structure which stores a set of strings. A node in the trie corresponds to a prefix of one or multiple strings.

4.3 Fuzzy and Tandem Cosine Distance

An approach was proposed where a semi-metric search (Chap. 5) using the MVP-tree was employed to speed up the search in a database of theoretical spectra [114]. Two variants of cosine distance were used as mass spectra similarity functions – the fuzzy cosine distance and tandem cosine distance. The approach is implemented in the application MSFound as a part of the framework MoBIoS [115].

4.3.1 Fuzzy Cosine Distance

The definition of the fuzzy cosine distance requires a high-dimensional boolean representation of mass spectra (Fig. 6.1). For example, let the range of $\frac{m}{z}$ values in a mass spectrum be 0-2,000 Da and let it be divided in subintervals of 0.1 Da. Then the spectrum is represented by a 20,000-dimensional boolean feature vector

having ones at places corresponding to intervals for which the $\frac{m}{z}$ value is nonzero in the spectrum.

Given two boolean vectors $\vec{A}$, $\vec{B}$ and the $\frac{m}{z}$ error peak tolerance τ, the *shared peak count* (SPC) is defined by Eq. 4.1.

$$SPC_\tau(\vec{A},\vec{B}) = \sum_i match(a_i,b_j); j \in [i-\tau, i+\tau]; a_i \in \vec{A}; b_j \in \vec{B} \tag{4.1}$$

$$match(a_i,b_j) = \begin{cases} 1; & a_i = b_j = 1; match(a_m,b_j) = 0, m \in [1,i) \\ 0; & otherwise \end{cases} \tag{4.2}$$

The Eq. 4.2 counts two peaks as equal when they lie within $\pm\tau$ bins of each other. The condition $match(a_m,b_j) = 0, m \in [1,i)$ ensures that each peak is matched with another peak exactly once, i.e., multiple matches of the same peak with other peaks are not counted. When $\tau = 0$, the Eq. 4.1 is reduced to the dot product on boolean vectors $\vec{A}$ and $\vec{B}$ (Eq. 4.3).

$$SPC_{\tau=0}(\vec{A},\vec{B}) = \sum_i match(a_i,b_i) = \vec{A}\cdot\vec{B} \tag{4.3}$$

The cosine similarity is defined as the normalized dot product between two vectors (Eq. 5.9). The fuzzy cosine similarity $\cos_\tau$ is defined as the normalized SPC_τ (Eq. 4.4), where $\|\ldots\|$ is the L_2 norm of a vector, $a_i \in \vec{A}$ and $b_i \in \vec{B}$.

$$\cos_\tau(\vec{A},\vec{B}) = \frac{SPC_\tau(\vec{A},\vec{B})}{\left\|\vec{A}\right\|\left\|\vec{B}\right\|} \tag{4.4}$$

Finally, the fuzzy cosine distance is defined as the inverse of $\cos_\tau$ (Eq. 4.5).

$$d_{fuzzy}(\vec{A},\vec{B}) = \arccos(\cos_\tau(\vec{A},\vec{B})) \tag{4.5}$$

d_{fuzzy} is a semi-pseudometric because it violates the reflexivity and triangle inequality (Sec. 5.1.1), what is caused by the $\frac{m}{z}$ error tolerance $\tau > 0$. The d_{fuzzy} has high intrinsic dimensionality ρ (Sec. 5.1.8) and it cannot be efficiently utilized by metric access methods (MAMs).

4.3.2 Tandem Cosine Distance

Let $S_A = \{\vec{A}, M_A\}$ be a mass spectrum where $\vec{A}$ is the high-dimensional boolean vector and M_A is the precursor mass of the spectrum S_A. The tandem cosine distance d_{tandem} is defined by Eq. 4.7. d_{tandem} is a linear combination of d_{fuzzy} with the difference of precursor masses $d_{precursor}$ of compared spectra (Eq. 4.6). λ is the precursor mass error tolerance. c_1 and c_2 are weights of the members in the equation (e.g., $c_1 = c_2 = 1$). Since $d_{precursor}$ is the L_1 distance computed over one-dimensional vectors and d_{tandem} is a linear combination of d_{fuzzy} with $d_{precursor}$, d_{tandem} is also a semi-pseudometric. d_{tandem} has very low ρ and it can be efficiently utilized by MAMs. A disadvantage of d_{tandem} is that the support of modifications in query mass spectra has not been discussed.

$$d_{precursor}(M_A,M_B) = \begin{cases} 0; & |M_A - M_B| \le \lambda \\ |M_A - M_B|; & otherwise \end{cases} \tag{4.6}$$

$$d_{tandem}(S_A,S_B) = c_1 d_{fuzzy}(\vec{A},\vec{B}) + c_2 d_{precursor}(M_A,M_B) \tag{4.7}$$

4.3.3 MVP-tree

The vantage point tree (VP-tree) is a simple metric access method (say, a predecessor of M-tree (Sec. 5.1.7)) which splits indexed objects into subsets pruned during the searching [116] [117]. In a binary VP-tree, each internal node has the form of $(V, d_M, L_{ptr}, R_{ptr})$ where V is a vantage point (pivot), d_M is a median distance among the distances from V to all objects indexed below this node, L_{ptr} and R_{ptr} are pointers to the subtrees. The multi vantage point tree (MVP-tree) is an extension of the VP-tree. In contrast to the VP-tree, the MVP-tree uses two pivots in each node. Every node of the MVP-tree can be seen as two levels of the VP-tree where all the children nodes at the lower level use the same pivot. Thus the MVP-tree stores less pivots in non-leaf levels than the VP-tree.

4.3.4 Semi-metric Search

Even though d_{tandem} violates the triangle inequality and the search using MVP-tree is approximate, the violation of the triangle inequality can be controlled. Lets assume a metric distance d, a query object q, a pivot p and a database object o. The triangle inequality is violated when $d(q,p) + d(o,p) < d(q,o)$. If there is a κ such that $d(q,p) + d(o,p) + \kappa \geq d(q,o)$, we say d violates the triangle inequality by this amount κ. Let κ_u be an upper bound on κ, a pruning condition in a metric range query with a radius r can be adjusted or fixed to return exact results by κ_u (Eq. 4.8) [114][118].

$$|d(q,p) - d(o,p)| > r + \kappa_u \tag{4.8}$$

4.4 Locality Sensitive Hashing

An approach for speeding up the search in theoretical mass spectra databases based on the locality sensitive hashing (LSH) was also proposed [119][120][121]. The LSH is an approximate algorithm which uses a hash-based indexing structure and the properties of metric spaces (Chap. 5) to obtain fast search times despite the high intrinsic dimensionality ρ (Sec. 5.1.8). The method provides good probabilistic bounds on the error of returned results [122][123][124].

The basic idea is to design a set of hash functions to pre-process the theoretical spectra in the database so that for each query spectrum the set of hash functions can be used to find theoretical spectra closest to the query spectrum. Let's assume a random spectrum and the cosine similarity (Eq. 5.9) as a hash function. Any query spectrum which is similar to the theoretical spectrum will have similar score to the random spectrum.

To implement this idea, a spectrum must be represented as a high-dimensional boolean vector (Fig. 6.1). An efficient filtering can be then achieved by ANN (approximate nearest neighbor) [125] queries but the efficiency is guaranteed only when L_1 (Eq. 5.6) or L_2 (Eq. 5.7) distances are used. The approach approximates the cosine distance by L_2 distance under the assumption that $1 - \cos(\vec{A}, \vec{B}) \approx L_2(\vec{A}, \vec{B})$ for small angles where $\vec{A}$, $\vec{B}$ are high-dimensional boolean vectors. A drawback of LSH is that a method for identification of peptide sequences from spectra containing modifications has not been proposed.

4.4.1 Family of Hash Functions

A family H of functions $h : \mathbb{R}^n \to U$ is called (p_1, p_2, r, cr)-sensitive, if for any high-dimensional object $\vec{O}$ and query $\vec{Q}$:

- if $\left\|\vec{O} - \vec{Q}\right\| \leq r$ then $Pr[h(\vec{O}) = h(\vec{Q})] \geq p_1$
- if $\left\|\vec{O} - \vec{Q}\right\| \geq cr$ then $Pr[h(\vec{O}) = h(\vec{Q})] \leq p_2$

where $p_1 > p_2$, r is the query radius, n is the dimension of space, U is the universe and $c > 1$ is the approximation factor [122].

Given a family H of hash functions, the LSH chooses t of them and concatenates them into a t-dimensional hash function $\vec{g_j}(\vec{A}) = (h_{1,j}(\vec{A}), \ldots, h_{t,j}(\vec{A}))$ to decrease the frequency of collisions. L different groups of hash functions are chosen uniformly and independently at random, i.e., $\vec{g_1}, \ldots, \vec{g_L}$. The parameters n and L are crucial to tune LSH because they amplify the gap between P_1 and P_2.

The LSH family (Eq. 4.9) has been proposed for the Euclidean space [126]. A random projection of $\mathbb{R}^n$ onto a 1-dimensional line is assumed. The line is chopped into segments of length w and shifted by a random value $b \in [0, w)$. The projection vector $\vec{P} \in \mathbb{R}^n$ is constructed by picking coordinates of $\vec{P}$ from the Gaussian distribution.

$$h(\vec{A}) = \left\lfloor \frac{\vec{P} \cdot \vec{A} + b}{w} \right\rfloor \tag{4.9}$$

4.4.2 Data Structure and Query Processing

The data structure is constructed by placing each point $\vec{O}$ from the database into buckets $\vec{g_j}(\vec{O})$ where $j = 1, \ldots, L$. When a query $\vec{Q}$ is being processed, all the buckets $\vec{g_1}(\vec{Q}), \ldots, \vec{g_L}(\vec{Q})$ are scanned and all the points from them are retrieved. Afterwards, their distances to $\vec{Q}$ are computed and any point is reported that is a valid answer to $\vec{Q}$ (Alg. 1) [122].

Algorithm 1: Locality Sensitive Hashing

Preprocessing:
begin
1. Choose L functions $\vec{g_j}$, $j = 1, \ldots, L$, by setting $\vec{g_j} = (h_{1,j}, h_{2,j}, \ldots h_{t,j})$ where $h_{1,j}, \ldots, h_{t,j}$ are chosen at random from the LSH family H;
2. Construct L hash tables, where for each $j = 1, \ldots, L$, the j^{th} hash table contains the dataset points hashed using the function $\vec{g_j}$;

Range Query:
begin
3. **for** $j = 1$ **to** L **do**
4. Retrieve the points from the bucket $\vec{g_j}(\vec{Q})$ in the j^{th} hash table;
5. For each of the retrieved points, compute the distance from $\vec{Q}$ to it and report the point if it is a correct answer;

4.5 Inverted Files

Several approaches have been proposed where inverted files are utilized to index a database of theoretical spectra [127][128][129][130]. The inverted index takes the advantage of the high sparsity of high-dimensional boolean vectors generated from the mass spectra (Fig. 6.1). Instead of storing a high-dimensional sparse boolean vector, a compact representation can be used where the positions of ones (i.e., dimensions in which the values are nonzero) are substituted with values of the compact vector. For example, the compact representation of the vector in Fig. 6.1 is $\vec{x} = \langle 7, 13, 18, 23, 27, 34 \rangle$.

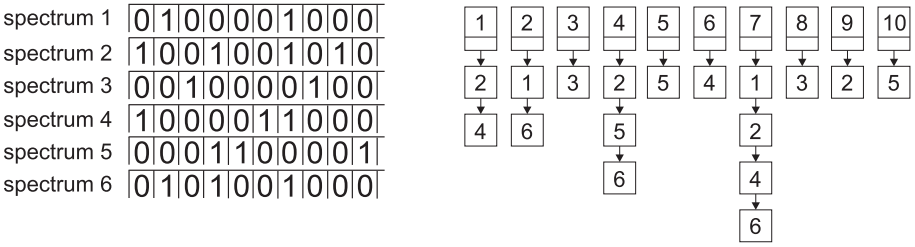


Figure 4.3: Inverted index

For each value in the compact representation, an inverted index stores a list of spectra where the value occurs (Fig. 4.3). When a query is being processed, the lists of the index are scanned in parallel. Each list has its own cursor pointing to an identifier of a spectrum. Only the cursor pointing to the smallest identifier can move forward. When the cursors are aligned, various boolean operations can be realized, e.g., an intersection (AND), a union (OR) or a complement (NOT). A common mass spectra similarity is the normalized dot product which is computed using the operation AND (Eq. 5.9). A problem can arise because lengths of the lists are not distributed evenly and the lists tend to be very long. Therefore, the high sparsity of boolean vectors is crucial for the efficiency of the inverted index.

A support of modifications in query mass spectra is not commonly mentioned in approaches based on the inverted index. However, recently, an approach has been proposed which supports also spectra with modifications [130]. Let $S = \{a_1, a_2, \ldots, a_n\}$ be a list of $\frac{m}{z}$ values where a_n is the precursor mass of the

Algorithm 2: Index Match Algorithm

Input: A spectrum $S = \{a_1, a_2, \ldots, a_n\}$ and a set of spectra $\tau = \{T_1, T_2, \ldots, T_k\}$ represented as a set of linked lists $L_1, L_2, \ldots, L_M$;
Output: A spectrum T in τ with the best mass counting score with S;

```
begin
1   Initialize score p_j = 0 for each spectra T_j ∈ τ;
2   for each mass a_i in S do
3       for each non-null node T_j in the linked list L_{a_i} do
4           Increase p_j by 1 ;
5   Output spectrum T_j with the highest score p_j;
```

Algorithm 3: Index Diagonal Algorithm

Input: A spectrum $S = \{a_1, a_2, \ldots, a_n\}$ and a set of spectra $\tau = \{T_1, T_2, \ldots, T_k\}$. All sub-vectors of the spectra in τ are represented as a set of linked lists $L_1, L_2, \ldots, L_M$;
Output: A spectrum T in τ with the best diagonal score with S;

begin
1 **for** *each sub-vector* $S[i..n]$ *of* S, $i = 1, \ldots, n-2$ **do**
2 Use Alg. 2 to find the best scoring sub-vector from τ (the sub-vector and $S[i..n]$ have the best mass counting score);
3 Compare the best scoring sub-vectors for $S[1..n], \ldots, S[n-2..n]$ and output the best one and its corresponding spectrum;

Algorithm 4: Two-Index Diagonal Algorithm

Input: A spectrum $S = \{a_1, a_2, \ldots, a_n\}$ and a set of spectra $\tau = \{T_1, T_2, \ldots, T_k\}$. All left sub-vectors and right sub-vectors of the spectra in τ are represented as two sets of linked lists;
Output: A spectrum T in τ with the best diagonal score with S;

begin
1 **for** $i = 1$ **to** $n-1$ **do**
2 Use Alg. 2 to find the sub-vector pair T[j..0] and T[j..m] such that C(S[i..0],T[j..0]) + C(S[i..n],T[j..m]) is maximized;
3 Compare the best scoring sub-vector pairs for $i = 1, \ldots, n-1$ and output the best scoring one and its corresponding spectrum;

spectrum. Let $s_1 s_2 \ldots s_N$ be a high-dimensional boolean representation of the spectrum S where $N = a_n$. If there is a mass i in the spectrum S and $i \neq n$, $s_i = 1$; otherwise, $s_i = 0$. For example, a spectrum $\{2, 4, 7, 10\}$ is represented by a vector 0101001000.

For a pair of spectra $S = \{a_1, a_2, \ldots, a_n\}$ and $T = \{b_1, b_2, \ldots, b_m\}$, a mass pair (a_i, b_j) is matched if $a_i = b_j$, $a_i \neq a_n$ and $b_j \neq b_m$. The number of matched mass pairs (i.e., the dot product over the high-dimensional vectors) is called mass counting score and denoted $C(S, T)$.

Let $\tau = \{T_1, T_2, \ldots, T_k\}$ be a set of theoretical mass spectra. For simplicity, let's assume that all the spectra in τ have the same precursor mass M. A linked list is generated for each column of aligned high-dimensional boolean vectors (Fig. 4.3). Having a set of linked lists $L_1, L_2, \ldots, L_M$, the index match algorithm is proposed to find a pair of spectra (S, T) with the maximum $C(S, T)$ where $T \in \tau$ (Alg. 2).

The diagonal score $D(S, T)$ has been proposed for spectra with modifications. When a spectrum contains a modification, $\frac{m}{z}$ values are shifted and thus ones in its high-dimensional representation are also shifted. Let $T(u)$ be a spectrum generated by shifting each mass in T by u, that is $T(u) = \{b_1 + u, b_2 + u, \ldots, b_m + u\}$. The diagonal score $D(S, T) = \max_u C(S, T(u))$ is the maximum mass counting score among all shift values.

The index diagonal algorithm has been proposed to compute the diagonal score (Alg. 3). The sub-vectors of theoretical spectra in τ and sub-vectors of the query spectrum S are generated for this purpose. For example, for each '1' in the high-dimensional boolean representation of S except the last, all elements left to the '1' are removed (including itself) from the vector to generate a sub-vector of S. The sub-vector generated from the i^{th} '1' is denoted $S[i..n]$. In total, $n-2$ sub-vectors are generated from S. For example, the spectrum $\{2, 4, 7, 10\}$ having the boolean representation 0101001000 has two sub-vectors 01001000 and 001000. The sub-vectors of S and linked lists formed from sub-vectors of spectra in τ are used in Alg. 3 to compute the diagonal scores. A prove that Alg. 3 reports the diagonal scores correctly is shown in [130].

A modification of the diagonal algorithm has been also proposed where two inverted indexes are used – one for left sub-vectors of spectra in τ and the other for right sub-vectors of spectra in τ (Alg. 4). The *right* sub-vectors are the same as above defined sub-vectors. A *left* sub-vector is the reverse of the sub-vector left to the element '1'. For example, the spectrum $\{2, 4, 7, 10\}$ having the boolean representation 0101001000 is split by third '1' into a left (reversed) sub-vector '001010' and a right sub-vector '000'. In total, the spectrum has three (in general, $n-1$) different pairs of sub-vectors (0,01001000), (010,001000) and (001010,000).

4.6 Other approaches

A vast number of approaches have been proposed to speed up the search in mass spectra databases. For example, there is an approach based on the hashing of similar spectra into 64-bit integers [131]. A group of approaches has been proposed where the machine learning techniques like support vector machines or neural networks are utilized [132] [133] [134]. For example, the neural network approach is based on a self-organizing map (SOM). The theoretical spectra are converted to high-dimensional vectors and the SOM is trained. For a query spectrum, a multi-point range query is performed using SOM, a candidate set of peptide sequences is obtained and then a scoring function is applied.

There is also an approach which indexes the database of protein/peptide sequences by suffix trees [135] [136]. On the other hand, a graph algorithm is used to pre-process a query mass spectrum. The suffix tree is then searched for candidate peptide sequences against the spectrum graph. The correct peptide sequence is then selected by a scoring/similarity function. Another approach is based on longest common prefixes and suffix arrays [137].

Other approaches are based on parallelization [138], GPU processing [139] or hardware acceleration of previously proposed methods [140]. Some approaches have been proposed which utilize a combination of algorithmic and software engineering techniques (say, a set of cheats) to speed up the identification of peptides using SEQUEST-like scoring [58][59]. An overview of other methods based on the database indexing, database reduction and database splitting has been also proposed in [121].

Chapter 5

Metric and Non-metric Access Methods

Since the proposed approach for peptide sequences identification (Chap. 6) is based on metric and non-metric (or semi-metric) similarity search, we need to briefly summarize the main points concerning metric access methods (MAMs) [116] and non-metric access methods [141].

5.1 Metric Access Methods

The MAMs were designed for efficient similarity search in multimedia databases where a pair-wise metric distance $d(x, y)$ is used. In the literature, there is a vast number of MAMs based on distance matrix methods, tree-based indexes, hashed indexes and index-free MAMs [116].

We define the basic properties of metric spaces and we give a brief overview of several metric distances. Then the LAESA distance matrix method and the M-tree are described. Finally, performance estimation methods and cost measures are briefly mentioned.

5.1.1 Metric Space and Metric Distance

Let D be a domain of objects and let d be a metric distance where $d : D \times D \to \mathbb{R}$. The *metric space* M is a pair $M = (D, d)$. The metric distance d determines the similarity between any two objects x, y and satisfies the following postulates:

1. **Reflexivity**
$$\forall x \in D, d(x, x) = 0, \tag{5.1}$$
2. **Non-negativity**
$$\forall x, y \in D, x \neq y \Rightarrow d(x, y) > 0, \tag{5.2}$$
3. **Symmetry**
$$\forall x, y \in D, d(x, y) = d(y, x), \tag{5.3}$$
4. **Triangle inequality**
$$\forall x, y, z \in D, d(x, z) \leq d(x, y) + d(y, z). \tag{5.4}$$

The metric postulates (especially the triangle inequality) are crucial to organize database objects within metric regions of MAMs and to prune irrelevant regions of MAMs while searching [116]. In case the triangle inequality is partially violated, the distance d is called the semi-metric distance. When the reflexivity is not satisfied the d is called pseudo-metric.

5.1.2 Minkowski Distances

The well known group of metric distances are Minkowski distances L_p which are suitable when objects are represented by vectors of constant sizes. The L_p distances can be summarized by Eq. 5.5 where $\vec{x} = (x_1, ..., x_n)$ and $\vec{y} = (y_1, ..., y_n)$.

$$L_p(\vec{x}, \vec{y}) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}; p \geq 1; x_i, y_i \in \mathbb{R} \tag{5.5}$$

Manhattan Distance

Manhattan distance L_1 is a special case of the L_p distance where $p = 1$ and is defined by Eq. 5.6.

$$L_1(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{5.6}$$

Euclidean Distance

Likely, the most known metric distance from the group of Minkowski distances is the Euclidean distance L_2 which is defined by Eq. 5.7.

$$L_2(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{5.7}$$

Maximum Distance

The maximum distance L_∞ is a variant of the L_p distance where $p = \infty$ (Eq. 5.8). L_∞ is used in LAESA metric access method to determine a lower-bound of a distance between a query object and a database object (Sec. 5.1.6).

$$L_\infty(\vec{x}, \vec{y}) = \max_{i=1}^{n} |x_i - y_i| \tag{5.8}$$

5.1.3 Cosine Similarity

Cosine similarity (or normalized dot product) determines the cosine of an angle φ between vectors $\vec{x}$ and $\vec{y}$ in the n-dimensional space (Eq. 5.9), where $\|\ldots\|$ is the *Euclidean L_2 norm*. The cosine similarity satisfies only the symmetry. Since the similarity between objects is bigger with bigger $\cos\varphi$, the cosine similarity does not satisfy the reflexivity. Since $\cos\varphi \in \langle -1, 1 \rangle$ the postulate of non-negativity is not satisfied. The triangle inequality is also violated what can be proven by a counterexample. Let's assume $\vec{x} = \{0, 1, 1\}$, $\vec{y} = \{0, 1, 0\}$ and $\vec{z} = \{1, 0, 0\}$, then

$\cos(\vec{x},\vec{y}) = \frac{1}{\sqrt{2}}$, $\cos(\vec{x},\vec{z}) = 0$ and $\cos(\vec{y},\vec{z}) = 0$. Since $0 + 0 \ngeq \frac{1}{\sqrt{2}}$, the triangle inequality is violated.

$$\cos\varphi = \cos(\vec{x},\vec{y}) = \frac{\vec{x}\vec{y}}{\|\vec{x}\|\,\|\vec{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{5.9}$$

The cosine similarity can be turned into the angle distance $d_{angle}(\vec{x},\vec{y})$ by the arccos function (Eq. 5.10). The angle distance is a pseudo-metric what means that the metric postulates are satisfied, but it may happen that $d_{angle}(\vec{x},\vec{y}) = 0$ for $\vec{x} \neq \vec{y}$. d_{angle} gives the size of an angle φ between the vectors $\vec{x}$ and $\vec{y}$ where $\varphi \in \langle 0, \frac{\pi}{2} \rangle$.

$$d_{angle}(\vec{x},\vec{y}) = \varphi = \arccos\left(\frac{\vec{x}\vec{y}}{\|\vec{x}\|\,\|\vec{y}\|}\right) \tag{5.10}$$

5.1.4 Hausdorff Distance

The Hausdorff distance d_H [116] is a metric distance defined on sets of objects A and B (Eq. 5.12). The d_H is computed as follows:

1. For each object in the set A, the distance to its nearest neighbor in the set B is determined.
2. The maximum distance from the distances to the nearest neighbors is selected.
3. The steps 1 and 2 are repeated with the A and B switched.
4. The maximum distance from the two maximum distances forms the result.

The ground distance d_X operating on the objects of A, B can be any other distance (e.g., L_2 in case of point sets). If d_X is a metric distance then d_H is also a metric distance. The time complexity to compute the d_H is $O(nm)\cdot O(d_X(\cdot,\cdot))$ where n and m are the numbers of objects in A and B, and $O(d_X(\cdot,\cdot))$ is the time complexity of the ground distance d_X.

$$h(A,B) = \max_{a\in A}\left\{\min_{b\in B}\{d_X(a,b)\}\right\} \tag{5.11}$$

$$d_H(A,B) = \max(h(A,B), h(B,A)) \tag{5.12}$$

5.1.5 Similarity Queries

The most common types of similarity queries are range queries $\mathrm{R}(q,r)$ and k nearest neighbor queries $\mathrm{kNN}(q,k)$ where q is a query object and r is a query radius (Fig. 5.1). When a sequential scan of a database $DB \subseteq D$ is employed, the distance $d(q,o)$ must be computed between the query object q and any database object $o \in DB$.

In case of the range query, the sequential scan of the database is performed while the objects for which $d(q,o) \leq r$ are appended to a result set (Eq. 5.13).

In case of the kNN query, a priority queue PQ of k nearest neighbors is updated while $d(q, o)$ are being computed (Eq. 5.14). The result of the kNN query may depend on the order of objects in DB and on the implementation of the kNN query when more objects are exactly in the same distance. For example, in Fig. 5.1 observe that the k^{th} and $(k+1)^{th}$ objects are in the same distance ($k = 3$).

MAMs employ the metric postulates (especially the triangle inequality) to organize the objects in DB into metric regions and to prune these regions while searching. When the range and kNN queries are performed on a MAM, many of the distances $d(q, o)$ are not computed and thus the search is significantly faster.

$$R(q, r) = \{o \in DB, d(o, q) \leq r\} \tag{5.13}$$

$$kNN(q, k) = \{PQ \subseteq DB, |PQ| = k \wedge \forall_{x \in PQ,\ y \in DB-PQ} : d(q, x) \leq d(q, y)\} \tag{5.14}$$

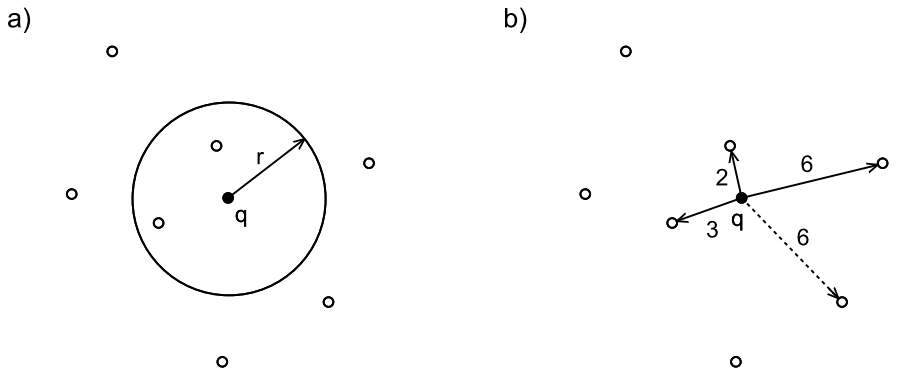


Figure 5.1: Similarity queries – (a) range query and (b) kNN query

5.1.6 LAESA

Pivot tables (or distance matrix methods) represent a simple but efficient solution to the similarity search. A representative of pivot tables is LAESA (Linear Approximating and Eliminating Search Algorithm) [142] [143] what is an improvement of AESA [144]. While AESA has quadratic space-complexity, LAESA has the complexity only linear.

In general, a set P of objects (so-called pivots) is selected from the database. A vector of distances between a database object $o_i \in DB$ and all the pivots $p_j \in P$ is pre-computed $\{d(o_i, p_1), \ldots, d(o_i, p_j), \ldots, d(o_i, p_l)\}$ where l is the number of pivots. The vectors of all database objects form a distance matrix – the pivot table.

When performing a query q, a vector of distances between q and the pivots is pre-computed the same way as for a database object o_i, i.e., $\{d(q, p_1), \ldots, d(q, p_j), \ldots, d(q, p_l)\}$. A query corresponds to the sequential scan of the entire database with the difference that a lower-bound lb of a distance between q and o_i is determined from the precomputed distances as $lb = \max_{p_j \in P}\{|d(q, p_j) - d(o_i, p_j)|\}$. In fact, the max says that L_∞ distance (Eq. 5.8) is computed. The lb allows to

prune irrelevant objects while the distance $d(q, o_i)$ is not computed in many cases and thus the search is faster than the sequential scan.

To show the advantage of LAESA, the $d(\cdot, \cdot)$ must be a time-consuming distance. Since L_∞ is used to determine the lb, the LAESA is not efficient with L_p distances. When n-dimensional objects are indexed and $l \geq n$, a computation of lb between the query q and any database object o_i is more time-consuming then a direct computation of $d(q, o_i)$.

Another drawback is that the entire matrix is scanned during the query processing and thus the LAESA is efficient only when the entire matrix is stored in the main memory. The problem is that the matrix may be very space-consuming (depending on the number of pivots l and the number of object in DB).

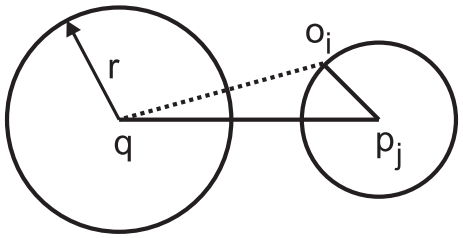


Figure 5.2: LAESA – filtering using the lower-bound

Range Query

The range query algorithm $R(q, r)$ on LAESA is shown in Alg. 5. In fact, the sequential scan of the database is performed while for each database object o_i, the

Algorithm 5: LAESA – range query

Input:
Database of objects $\mathsf{o}_i \in \mathsf{DB}$, query object q, radius r, set of pivots $\mathsf{p}_j \in \mathsf{P}$, metric distance d, matrix of pre-computed distances $\mathsf{d}(\mathsf{o}_i, \mathsf{p}_j)$.
Output:
Set of objects $\mathsf{S} \subseteq \mathsf{DB}$ where $\forall_{\mathsf{o}'_i \in \mathsf{S}} : \mathsf{d}(\mathsf{q}, \mathsf{o}'_i) \leq \mathsf{r}$.
begin
    $\mathsf{S} = \emptyset$;
    for *all* $\mathsf{p}_j \in \mathsf{P}$ **do**
        compute $\mathsf{d}(\mathsf{q}, \mathsf{p}_j)$;
    for *all* $\mathsf{o}_i \in \mathsf{DB}$ **do**
        $\mathsf{lb} = \max_{\mathsf{p}_j \in \mathsf{P}} \{ |\mathsf{d}(\mathsf{q}, \mathsf{p}_j) - \mathsf{d}(\mathsf{o}_i, \mathsf{p}_j)| \}$;
        if $\mathsf{lb} \leq \mathsf{r}$ **then**
            compute $\mathsf{d}(\mathsf{q}, \mathsf{o}_i)$;
            if $\mathsf{d}(\mathsf{q}, \mathsf{o}_i) \leq \mathsf{r}$ **then**
                append o_i to S;
    return S;

lb is computed. When $lb \leq r$, the $d(q, o_i)$ must be computed. When $d(q, o_i) \leq r$, the o_i is appended to a result set. The principle of the filtering using the lower-bound is shown in Fig. 5.2. The task is to determine whether the o_i is inside the query ball (q, r). The $d(q, p_j)$ is known because it is computed when the range query is initialized. The $d(o_i, p_j)$ is known because the value is stored in the pivot table. The $d(q, o_i)$ is unknown. The $d(q, o_i)$ does not have to be computed because its lower-bound $|d(q, p_j) - d(o_i, p_j)|$ is larger than r. So, the o_i surely cannot be in the query ball (q, r) and thus o_i is ignored.

kNN Query

The k nearest neighbor query $kNN(q, k)$ on LAESA is shown in Alg. 6. First, the lb is computed for all objects in the database. Second, the database is sorted in ascending order w.r.t. lb. Third, the database sorted by lb is scanned and

Algorithm 6: LAESA – kNN query

```
1  Input:
2  Database of objects o_i ∈ DB, query object q, number of nearest neighbors
   k, set of pivots p_j ∈ P, metric distance d, matrix of pre-computed distances
   d(o_i, p_j).
3  Output:
4  Priority queue PQ of k nearest neighbors (priority is determined by d(·,·)).
5  begin
6      PQ = ∅;
7      for all p_j ∈ P do
8          compute d(q, p_j);
9      for all o_i ∈ DB do
10         lb_(q,o_i) = max_{p_j ∈ P} {|d(q, p_j) − d(o_i, p_j)|};
11     for all o_i ∈ DB sorted in ascending order w.r.t. lb_(q,o_i) do
12         if number of objects in PQ < k then
13             compute d(q, o_i);
14             insert o_i with d(q, o_i) into PQ;
           else
15             /* d(q, o_k) is the distance between q and object in PQ
                  with the smallest priority (k^th nearest neighbor)
                  */
16             if d(q, o_k) < lb_(q,o_i) then
17                 return PQ;
18             compute d(q, o_i);
19             if d(q, o_i) < d(q, o_k) then
20                 insert o_i with d(q, o_i) into PQ;
21                 remove the object with the smallest priority from PQ to
                   keep only the k nearest neighbors;
22     return PQ;
```

a priority queue PQ of k nearest neighbors is being updated. When $d(q, o_k) < lb_{(q,o_i)}$ where o_k is the k^{th} nearest neighbor in PQ, the filtering terminates. For a large database, the sorting by $lb_{(q,o_i)}$ may be a time-consuming bottleneck. To prevent this behavior, the pivot table can be split into smaller blocks where the number of items in a block is an order of magnitude smaller than the number of objects in the database. The kNN query is then performed on all blocks where objects in a block are sorted by $lb_{(q,o_i)}$. When $d(q, o_k) < lb_{(q,o_i)}$ the scanning of a block terminates and the next block is processed.

5.1.7 M-tree

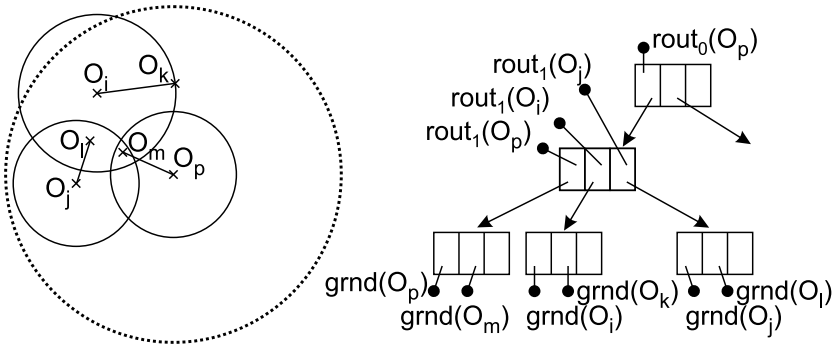


Figure 5.3: M-tree

The metric tree (M-tree) [145] is a hierarchical, dynamic (updatable) and balanced index structure that provides good performance in secondary memory, i.e., in database environments. The n-dimensional data objects are partitioned among the ball-shaped regions. The structure of M-tree consists of inner and leaf nodes. All indexed objects are stored in leaf nodes and duplicates of some of these objects are also used as centers of ball-shaped regions in the inner nodes.

While inner nodes contain *routing entries* associated with metric regions, leaf nodes are represented by *ground entries* containing data objects and identifiers uniquely identifying the data (Fig. 5.3). A routing entry (Eq. 5.15) contains the center of the ball-shaped region o_i, the radius of the region r_{o_i}, the pointer $ptr(T(o_i))$ to the covering subtree $T(o_i)$ and the pre-computed distance $d(o_i, o_p)$ between the center of the region and the center of the parent region o_p.

$$rout(o_i) = [o_i, r_{o_i}, ptr(T(o_i)), d(o_i, o_p)] \tag{5.15}$$

A ground entry (Eq. 5.16) consists of the indexed object o_i, the identifier of an externally stored database entry $oid(o_i)$ and the pre-computed distance $d(o_i, o_p)$ between the indexed object o_i and the center of the parent region o_p.

$$grnd(o_i) = [o_i, oid(o_i), d(o_i, o_p)] \tag{5.16}$$

For all regions in the covering subtree $T(o_i)$ of a region (o_i, r_{o_i}), the *nesting condition* must be satisfied (Eq. 5.17). Another words, the subregions of (o_i, r_{o_i}) may overhang of (o_i, r_{o_i}), but the nesting condition ensures that all indexed objects in the subregions are inside the region (o_i, r_{o_i}).

$$\forall_{o_j \in T(o_i)} : d(o_i, o_j) \leq r_{o_i} \tag{5.17}$$

Pruning Conditions

The M-tree employs two pruning conditions to filter out the irrelevant regions during the searching. Let (q, r_q) be a query region, the following conditions are used to prune irrelevant regions of M-tree.

1. **Basic filtering.** When $d(o_i, q) > r_{o_i} + r_q$ then for each $o_j \in T(o_i)$: $d(o_j, q) > r_q$ and thus the covering subtree $T(o_i)$ does not have to be searched (Fig. 5.4a).

2. **Parent filtering.** When $|d(o_p, q) - d(o_p, o_i)| > r_{o_i} + r_q$ where o_p is the center of the parent region of o_i, then $d(o_i, q) > r_{o_i} + r_q$ and thus the covering subtree $T(o_i)$ does not have to be searched (Fig. 5.4b).

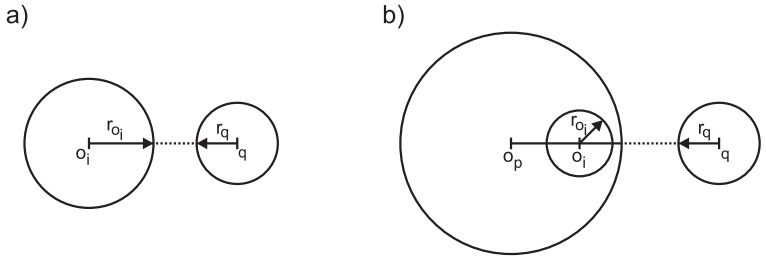


Figure 5.4: Pruning conditions in M-tree

Construction

The M-tree can be constructed by a statical or dynamical way. The statical way (so-called bulk-loading) is suitable when a large collection of objects is available in the time of construction. The algorithm clusters similar objects and then creates the tree hierarchy. The details of this approach are proposed in [116] [146].

When a large collection of objects is not available in the time of construction or when the database is frequently updated, the M-tree can be constructed dynamically by the insertion of single objects into an existing tree hierarchy. We briefly describe the dynamical construction of M-tree.

Algorithm 7: M-tree – construction (part 1)

```
1  Function InsertObject(T, o_i)
2  Input:
3  Root node of M-tree T, inserted object o_i.
4  Output:
5  None.
6  begin
7      N = FindLeaf(T, o_i);
8      if N is not full then
9          store o_i in the leaf node N;
       else
10         SplitNode(N, o_i);
```

Algorithm 8: M-tree – construction (part 2)

```
1  Function FindLeaf(N, o_i)
2  Input:
3  Node N, inserted object o_i.
4  Output:
5  Leaf node N.
6  begin
7      if N is a leaf node then
8          return N;
9      N' = {∀_{rout(o_j)∈N} : d(o_i, o_j) ≤ r_{o_j}};
10     if N' ≠ ∅ then
11         select o*_j such that rout(o*_j) ∈ N' : min{d(o*_j, o_i)};
       else
12         select o*_j such that rout(o*_j) ∈ N : min{d(o*_j, o_i) − r_{o*_j}};
13         r_{o*_j} = d(o*_j, o_i);
14     return FindLeaf(ptr(T(o*_j)), o_i);

15 Function SplitNode(N, o_i)
16 Input:
17 Node N, inserted object o_i.
18 Output:
19 None.
20 begin
21     N_all = {∀_{o_j} : o_j ∈ N} ∪ {o_i};
22     /* Select centers o_{p1}, o_{p2} of 2 new routing items from N_all  */
23     Promote(N_all, o_{p1}, o_{p2});
24     /* Split objects from N_all to N_{p1}, N_{p2} w.r.t. o_{p1}, o_{p2}  */
25     Partition(N_all, o_{p1}, o_{p2}, N_{p1}, N_{p2});
26     allocate a new node N';
27     store entries from N_{p1} into N and entries from N_{p2} into N';
28     if N is the root node then
29         allocate a new root node N_p;
30         store rout(o_{p1}) and rout(o_{p2}) in N_p;
       else
31         let rout(o_p) be the routing entry of N stored in the parent node N_p;
32         replace the entry rout(o_p) with rout(o_{p1}) in N_p;
33         if N_p is full then
34             SplitNode(N_p, o_{p2});
           else
35             store rout(o_{p2}) in N_p;
```

The M-tree structure can be constructed by insertions of single objects using the function *InsertObject* (Alg. 7). The function *FindLeaf* (Alg. 8) selects a leaf node in which a new object o_i will be stored. The leaf is selected whose center object o_j in its routing entry is closest to o_i. The entry is preferred for which an extension of r_{o_j} is not necessary. When such a region does not exist, the entry is selected for which a minimum extension of r_{o_j} must be done.

When the leaf node is selected, the object o_i (ground entry, respectively) is inserted into the leaf. In case the leaf node is overfilled, the node is split into two new nodes by the function *SplitNode* (Alg. 8). Two functions are called inside the *SplitNode* – *Promote* and *Partition*. The function *Promote* selects two centers o_{p_1} and o_{p_2} of newly created routing entries from the set of objects N_{all}. A few heuristics for an optimal selection of the centers are proposed in [116]. The function *Partition* splits the set of objects between the two newly created nodes. Again, a couple of heuristics have been developed for an optimal partitioning, e.g., an object can be inserted into the node whose center is closest to the object [116].

When the centers are selected and when the objects are partitioned, objects belonging to the center o_{p_1} are stored in the old node N and objects belonging to the center o_{p_2} are stored in a newly allocated node N'. When the old node N is the root of the M-tree, a new root must be allocated and routing items of o_{p_1} and o_{p_2} are stored in the newly allocated root. In case the old node N is not the root, the routing entry in its parent node N_p must be replaced with the routing entry of o_{p_1}. The routing entry of o_{p_2} must be also inserted into N_p. When o_{p_2} cannot be inserted into N_p because N_p is overfilled, the function *SplitNode* is called recursively on the parent node N_p.

The described algorithm is also known as the *single-way*. When an object is being inserted, the M-tree is traversed along exactly one branch and the first suitable leaf node is selected. A better resulting indexing structure (i.e., less and more tight clusters) can be achieved by the *multi-way* [116] [147]. Before inserting a new object o_i, a point query $\mathrm{R}(o_i, 0)$ is performed. For all visited leaves, the distances between o_i and the centers of their ball regions are computed. The leaf node whose pivot is the closest to o_i is then chosen to store the new object. If no such leaf is found, the single-way insertion is employed. This happens when no leaf node covers the area of o_i and the search algorithm terminates empty before reaching a leaf. A drawback is that the multi-way is more time-consuming than the single-way.

Range Query

When performing a range query (Alg. 9), the M-tree is scanned from the root, while the subtrees of the regions which overlap the query region must be searched as well, recursively. When a non-leaf node is processed, the pruning condition $|d(o_p, q) - d(o_i, o_p)| \leq r + r_{o_i}$ is used for a cheap elimination of regions which do not have to be scanned. $d(o_i, o_p)$ is stored in each routing entry and the value of $d(o_p, q)$ has been precomputed as $d(o_i, q)$ in the previous call of *RangeQueryIter*. When a region is not pruned by this condition, $d(o_i, q)$ must be computed and the overlap of the region (o_i, r_{o_i}) with the query region (q, r) is tested. When the region (o_i, r_{o_i}) overlaps (q, r), the *RangeQueryIter* is called recursively.

When a leaf node is processed, a simpler condition $|d(o_p, q) - d(o_i, o_p)| \leq r$ is used to filter out the objects which do not belong to a result set S. When an

object o_i is not eliminated by the condition, $d(o_i, q)$ must be computed and its presence in the query region (q, r) is tested. When the object o_i occurs in (q, r), o_i is appended to the result set S.

Algorithm 9: M-tree – range query

```
1  Function RangeQueryIter(N, q, r, S)
2  Input:
3  Node N, query object q, radius r, set of objects S.
4  Output:
5  Set of objects S.
6  begin
7      let o_p be the parent object of node N;
8      if N is not a leaf node then
9          for each rout(o_i) ∈ N do
10             if |d(o_p, q) − d(o_i, o_p)| ≤ r + r_{o_i} then
11                 compute d(o_i, q);
12                 if d(o_i, q) ≤ r + r_{o_i} then
13                     RangeQueryIter(ptr(T(o_i)), q, r, S);
       else
14         for each grnd(o_i) ∈ N do
15             if |d(o_p, q) − d(o_i, o_p)| ≤ r then
16                 compute d(o_i, q);
17                 if d(o_i, q) ≤ r then
18                     append o_i into S;
19     return S;

20 Function RangeQuery(T, q, r)
21 Input:
22 Root node of M-tree T, query object q, radius r.
23 Output:
24 Set of objects S where ∀_{o_i∈S} : d(q, o_i) ≤ r.
25 begin
26     S = ∅;
27     RangeQueryIter(T, q, r, S);
28     return S;
```

kNN Query

The implementation of the kNN query (Alg. 10 and Alg. 11) is not so straightforward as the implementation of the range query because the radius r is not known. At the beginning, the dynamic radius r is set to ∞ and during the query processing r is being reduced. At the end of the kNN query, r corresponds to the distance of k^{th} nearest neighbor. However, both kind of queries are processed with equal I/O costs (numbers of accessed nodes).

Algorithm 10: M-tree – kNN query (part 1)

```
1  Function kNNQuery(T, q, k)
2  Input:
3  Root node of M-tree T, query object q, number of nearest neighbors k.
4  Output:
5  Queue NN of k nearest neighbors.
6  begin
7      insert [T, −] into PR;
8      r = ∞;
9      for i = 1 to k do
10         NN[i] = [−, ∞];
11     while PR ≠ ∅ do
12         nextnode = ChooseNode(PR);
13         kNNSearchNode(nextnode, q, k, NN, PR, r);
14     return NN;
```

The kNN algorithm employs a priority queue PR (pending requests) to store unprocessed nodes and an array NN (nearest neighbors) of the length k. PR contains requests $[ptr(T(o_i)), d_{min}(T(o_i))]$ where $ptr(T(o_i))$ refers to an unprocessed subregion which cannot be pruned because it overlaps the dynamic query region (q, r). The priority of requests in PR is given by $d_{min}(T(o_i))$ what is a distance between a query q and a ball region (o_i, r_{o_i}) (Eq. 5.18). When the dynamic radius r is reduced during the query processing, the items for which $d_{min}(T(o_i)) > r$ are removed from PR.

$$d_{min}(T(o_i)) = \max\{0, d(o_i, q) - r_{o_i}\} \tag{5.18}$$

The array NN contains items $[o_i, d(q, o_i)]$ or $[-, d_{max}(T(o_i))]$ which are sorted according to their distances from q. The items $[-, d_{max}(T(o_i))]$ are presented in NN until k objects from leaves are revealed. $d_{max}(T(o_i))$ is the maximum distance between q and any object in the region (o_i, r_{o_i}) (Eq. 5.19). The function *NNUpdate* updates the sorted array NN and returns a new value of r which corresponds to $NN[k].d_{max}$.

$$d_{max}(T(o_i)) = d(o_i, q) + r_{o_i} \tag{5.19}$$

At the beginning of the KNN query, the PR is initialized with a request of the root node T. The PR is being processed in a loop until it is empty. In each iteration, a node with the minimum $d_{min}(T(o_i))$ is selected by *ChooseNode* and the node is processed by *kNNSearchNode*.

When a non-leaf node is being processed, the condition $|d(o_p, q) - d(o_i, o_p)| \leq r + r_{o_i}$ is used to eliminate regions which do not have to be scanned. The value of $d(o_p, q)$ is known because it is computed as $d(o_i, q)$ when the parent node is being processed. When a region is not pruned by this condition, $d(o_i, q)$ must be computed and the overlap of the region (o_i, r_{o_i}) with the dynamic query region (q, r) is tested. When the region (o_i, r_{o_i}) overlaps (q, r), NN and r are updated. When r is updated, all requests such that $d_{min}(T(o_i)) > r$ are removed from PR.

Algorithm 11: M-tree – kNN query (part 2)

```
1  Function ChooseNode(PR)
2  Input:
3  Priority queue of unprocessed nodes PR.
4  Output:
5  Node which will be processed.
6  begin
7      let d_min(T(o'_i)) = min{∀d_min(T(o_i)) ∈ PR};
8      remove entry [ptr(T(o'_i)), d_min(T(o'_i))] from PR;
9      return ptr(T(o_i'));

10 Function kNNSearchNode(N, q, k, NN, PR, r)
11 Input:
12 Node N, query object q, number of nearest neighbors k, queue of nearest
   neighbors NN, priority queue of unprocessed nodes PR, radius r.
13 Output:
14 None (NN, PR and r are updated).
15 begin
16     let o_p be the parent object of node N;
17     if N is not a leaf node then
18         for each rout(o_i) ∈ N do
19             if |d(o_p, q) − d(o_i, o_p)| ≤ r + r_{o_i} then
20                 compute d(o_i, q);
21                 if d_min(T(o_i)) ≤ r then
22                     append [ptr(T(o_i)), d_min(T(o_i))] to PR;
23                     if d_max(T(o_i)) ≤ r then
24                         r = NNUpdate(NN, [−, d_max(T(o_i))]);
25                     remove from PR all entries for which d_min(T(o_i)) > r;
       else
26         for each grnd(o_i) ∈ N do
27             if |d(o_p, q) − d(o_i, o_p)| ≤ r then
28                 compute d(o_i, q);
29                 if d(o_i, q) ≤ r then
30                     r = NNUpdate(NN, [o_i, d(o_i, q)]);
31                 remove from PR all entries for which d_min(T(o_i)) > r;
```

When a leaf node is being processed, the condition $|d(o_p, q) - d(o_i, o_p)| \leq r$ is used to filter out the objects which are not inside the dynamic query region (q, r). When an object o_i is not eliminated by the condition, $d(o_i, q)$ must be computed and its presence in (q, r) is tested. When o_i occurs in (q, r), NN and r are updated. Afterwards, unnecessary pending requests are removed from PR.

In the literature, many improvements of the M-tree have been proposed, e.g., the PM-tree (pivoting metric tree) which employs a global array of pivots to crop the regions of M-tree and to minimize the overlap of the regions with the query region [148] [149].

5.1.8 Performance Estimation

The requirement on metric postulates is crucial to index a database by MAMs. However, the postulates alone do not guarantee an efficient query processing. A few performance estimation methods have been proposed to estimate the efficiency of MAMs while any MAM implementation is not necessary for this purpose [141].

Intrinsic Dimensionality

The efficiency limits of any MAM heavily depend on the distance distribution in a database S, and can be formalized by the concept of *intrinsic dimensionality* ρ (Eq. 5.20), where μ is the mean and σ^2 is the variance of the distance distribution, d_i is the distance between any two objects in the database and n is the number of pairs of objects in the database [150].

$$\rho(S, d(\cdot, \cdot)) = \frac{\mu^2}{2\sigma^2}, \quad \mu = \sum_{i=1}^{n} \frac{d_i(\cdot, \cdot)}{n}, \quad \sigma^2 = \sum_{i=1}^{n} \frac{(\mu - d_i(\cdot, \cdot))^2}{n} \tag{5.20}$$

ρ is low (say, $\rho < 3$) if the data forms tight clusters. Hence, the database can be efficiently searched by a MAM, because a query region overlaps only a small number of clusters (we say that the indexability by MAMs is good). On the other hand, the high intrinsic dimensionality (say, $\rho > 10$) indicates that most of data objects are more or less equally far from each other. Hence, in intrinsically high-dimensional database there do not exist clusters and the search deteriorates to the sequential scan of the database (we say that the indexability is bad). The problem of high intrinsic dimensionality is, in fact, a generalization of the well-known *curse of dimensionality* into metric spaces [141] [150].

Distance Distribution Histogram

Beside ρ, a distance distribution histogram (DDH) [141] is commonly used as an indicator of the low or high intrinsic dimensionality (Fig. 5.5). The DDH shows a distribution of distances between any two objects in the database. For each distance $d \pm \delta$ where δ is an error caused by rounding, the distance frequency is plotted. d^+ is a maximum distance between objects. When a substantial part of the distribution is behind $\frac{d^+}{2}$ (say, on the right side of the DDH), the high ρ is indicated and thus the indexability by MAMs is bad. On the other hand, when the substantial part is in front of $\frac{d^+}{2}$ (say, on the left side), the low ρ is indicated and the indexability is good.

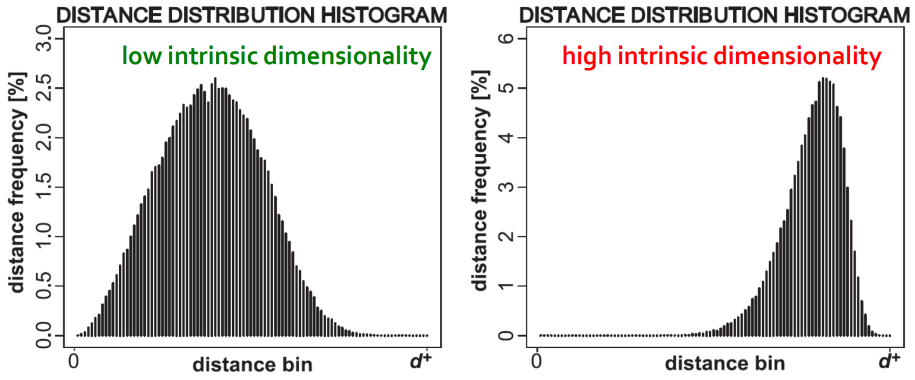


Figure 5.5: Distance distribution histograms

5.1.9 Cost Measures

When a MAM has been implemented, the following quantities can be used to evaluate the efficiency of the querying:

1. **Real time.** The time of a range or a kNN query is commonly compared to the time of the sequential scan of entire database to determine the speed up of querying on the MAM. A drawback is that the real time can be distorted depending on the implementation of the MAM.

2. **Number of distance computations.** The number of calls of the distance function d is counted. The number of distance computations is compared to the number of objects in the database (i.e., the number of distance computations of the sequential scan) to obtain the speed up. A disadvantage is that the measure is suitable only when d is a time-consuming distance, otherwise the results may be misleading.

3. **I/O costs.** The number of accessed nodes/blocks of a MAM is counted. The measure is suitable when the MAM cannot fit into the main memory and its blocks are transfered between the main memory and a HDD/SSD drive.

4. **Internal costs.** The number of any implementation-dependent operations is counted, e.g., when various auxiliary main-memory structures are used.

5.2 Non-Metric Access Methods

The metric postulates are commonly a restrictive criterion in general domains. However, a general similarity function can be converted to a distance and the fulfillment of metric postulates can be enforced to the distance. Hence, the MAMs can be correctly used to index and to search the database using the metric modification of the distance.

On the other hand, when a distance satisfies the metric postulates but the intrinsic dimensionality ρ is high, the triangle inequality can be partially violated and MAMs can be employed for fast but approximate search.

We briefly describe the enforcement of metric postulates into a non-metric distance. We define T-bases which control the enforcement of the triangle inequality and the retrieval error. Then the TriGen algorithm is described which automates a selection of the best T-base for a specified retrieval error. Finally, the NM-tree is defined as an extension of the M-tree for non-metric distances.

5.2.1 Enforcement of Metric Postulates

In many practical applications, a general similarity function is used but such a similarity cannot be commonly utilized by MAMs. Let's assume a bounded and normalized similarity function $s(\cdot,\cdot) \in \langle 0,1\rangle$ where a big value means that two compared objects are more similar than when s is a small value. The similarity s can be turned to a distance d by $d(\cdot,\cdot) = 1 - s(\cdot,\cdot)$. Even though d does not fulfill the metric postulates, their fulfillment can be fixed as follows [141] [151]:

1. **Reflexivity.** The reflexivity is not satisfied when $d(x,y) = 0$ where $x \neq y$. The problem can be fixed when a distance $d_R(x,y) = d(x,y) + \epsilon$ is used instead of $d(x,y)$ when $x \neq y$ where ϵ is a small positive constant.

2. **Non-negativity.** The non-negativity is satisfied because we assume that $d(x,y) \in \langle 0,1\rangle$.

3. **Symmetry.** The symmetry can be fixed, e.g., when a distance $d_S(x,y) = \max(d(x,y), d(y,x))$ is used instead of $d(x,y)$.

4. **Triangle inequality.** The triangle inequality can be added to the distance by utilization of a T-base (Sec. 5.2.3).

5.2.2 T-error

$$T\text{-}error = \frac{\#\ of\ non\text{-}triangular\ triplets}{\#\ all\ triplets} \tag{5.21}$$

Let's assume a semi-metric distance d (i.e., only the triangle inequality is violated) and let $\langle d(o_a,o_b), d(o_a,o_c), d(o_b,o_c)\rangle$ be a *triangle triplet* among the objects o_a, o_b, o_c, where $d(x,z) \leq d(x,y) + d(y,z)$, $x,y,z \in \{o_a,o_b,o_c\}$, $x \neq y \neq z$. The *T-error* (triangle error) is a ratio of triplets, which do not satisfy the triangle inequality, to all triplets in the database (Eq. 5.21).

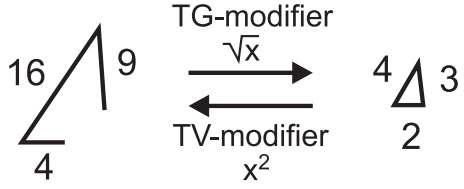


Figure 5.6: An example of TG-modifier and TV-modifier

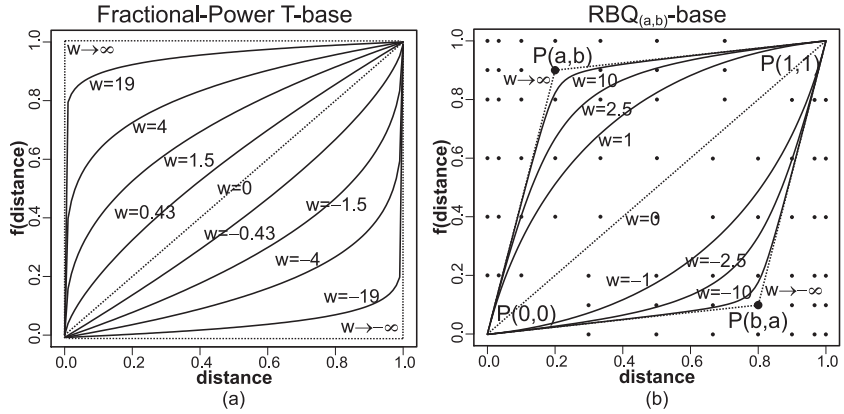


Figure 5.7: The FP-base and an $\text{RBQ}_{(a,b)}$-base

5.2.3 T-modifiers and T-bases

A T-modifier (triangle modifier) is a function $f(d(\cdot,\cdot))$, which is able to control the metricity (T-error, respectively) of the distance d. The T-modifier must not change the order of objects determined by d what is guaranteed when f is a monotonous function. A triangle generating modifier (TG-modifier) is a concave T-modifier which lowers the T-error (e.g., $f(x) = \sqrt{x}$). A triangle violating modifier (TV-modifier) is a convex T-modifier which increases the T-error (e.g., $f(x) = x^2$). A simple example is shown in Fig. 5.6.

A T-base $f(d(\cdot,\cdot), w)$ is a T-modifier with an additional parameter w, that aims to control the convexity or concavity of f. When $w > 0$, f gets more concave (a TG-modifier is used). When $w < 0$, f gets more convex (a TV-modifier is used). When $w = 0$, we get the identity $f(d(\cdot,\cdot), 0) = d(\cdot,\cdot)$. A simple T-base is the Fractional-Power base (FP-base) (Eq. 5.22), while a more sophisticated T-base is the Rational-Bézier-Quadratic base (RBQ-base) (Eq. 5.23). The behavior of the T-bases for different w is shown in Fig. 5.7. For a detailed description of the *rbq* function, see [141].

$$\text{FP}(d(\cdot,\cdot), w) = \begin{cases} d(\cdot,\cdot)^{\frac{1}{1+w}} & \text{for } w > 0 \\ d(\cdot,\cdot)^{1-w} & \text{for } w \leq 0 \end{cases} \tag{5.22}$$

$$\text{RBQ}_{(a,b)}(d(\cdot,\cdot), w) = \begin{cases} rbq(d(\cdot,\cdot), w, a, b) & \text{for } w > 0 \\ rbq(d(\cdot,\cdot), -w, b, a) & \text{for } w \leq 0 \end{cases} \tag{5.23}$$

5.2.4 TriGen

The TriGen algorithm was proposed to automate the search for an optimal weight w of a T-base [141]. The TriGen searches for a T-base f for which the ρ is minimized, while the T-error does not exceed a user-specified *T-error tolerance* θ. The modified distance $f(d(\cdot,\cdot), w)$ can be employed by any MAM for an exact but slower (T-error tolerance is zero, so ρ gets higher) or only an approximate but fast (T-error tolerance is positive, so ρ gets smaller) similarity search (metric or non-metric).

The TriGen algorithm is described in Alg. 12. A set τ of t triangle triplets is sampled from a sample of database objects S. For a T-base f^*, the T-error is

Algorithm 12: TriGen algorithm

```
1  Function TriGen(d, F, θ, S, iterLimit, t)
2  Input:
3  (Semi-)metric d, pool F of T-bases, T-error tolerance θ, database sample S,
   iteration limit iterLimit, number of sampled triplets t.
4  Output:
5  An optimal T-base f ∈ F and weight w.
6  begin
7      f = null;
8      w = 0;
9      maxIndexability = −∞;
10     τ = SampleTriplets(S, t, d);
11     for each f* ∈ F do
12         w* = 0;
13         w_best = null;
14         terror = ComputeTriangleError(f*, w*, τ);
15         /* TV- or TG- modification is selected                    */
16         if terror < θ then
17             w_lb = −∞;
18             w_ub = 0;
19             w* = −1;
           else
20             w_lb = 0;
21             w_ub = ∞;
22             w* = 1;
23         /* find the optimal weight w_best for a T-base f*         */
24         for i = 1 to iterLimit do
25             terror = ComputeTriangleError(f*, w*, τ);
26             if terror ≤ θ then
27                 w_best = w*;
28                 w_ub = w*;
               else
29                 w_lb = w*;
30             if w_lb = −∞ or w_ub = ∞ then
31                 w* = 2w*;
               else
32                 w* = (w_lb + w_ub)/2;
33         /* among the candidate T-bases, choose the one with the
               best ρ                                                */
34         if w_best ≠ null then
35             indexability = ComputeIndexability(f*, w_best, τ);
36             if indexability > maxIndexability then
37                 f = f*;
38                 w = w_best;
39                 maxIndexability = indexability;
40     return f and w;
```

computed using $w^* = 0$. If the T-error $< \theta$, a TV-modifier is searched. Otherwise, we are looking for a TG-modifier. Weights w_{lb}, w_{ub} and w^* are initialized. In a number of iterations specified by *iterLimit*, the T-error is computed and the weights are updated. If $w_{lb} = -\infty$ or $w_{ub} = \infty$, the weight w^* is doubled. When $w_{lb} \neq -\infty$ and $w_{ub} \neq \infty$, the interval between w_{lb} and w_{ub} is halved and an optimal w^* is searched. Finally, the T-base f with the best indexability is selected from a set of T-bases.

Although the TriGen algorithm allows to use MAMs also with non-metric distances, it does not guarantee that a particular non-metric distance modified into metric will be suitable for indexing by MAMs. In particular, a highly non-metric distance (exhibiting a high T-error) is modified by TriGen very aggressively to achieve the zero T-error, what means the resulting metric will imply high intrinsic dimensionality ρ making the database not indexable. Hence, when designing a new similarity that has to be indexable by MAMs, the attention should be given not only to the semantics of the similarity/effectiveness, but also to its indexability/efficiency (low both, the T-error and ρ).

5.2.5 NM-Tree

The NM-tree (non-metric tree) [152] is a modification of the M-tree which natively aggregates the TriGen algorithm to support the flexible approximate or exact search using an arbitrary (non)metric distance function. In the NM-tree, an input distance d is supposed to be a semi-metric (i.e., the triangle inequality is violated), while TriGen is applied before indexing in order to turn d into a metric d^{f_M} (i.e, $\theta = 0$). Distances stored in the NM-tree are always metric ones (i.e., $d^{f_M}(\cdot,\cdot)$). When a query (e.g., k-NN or range) is performed and the approximate search is required (i.e., $\theta > 0$), the distance d^{f_M} is by definition modified inversely by f_M^{-1} and then another modifier f' is applied, i.e., $f'(f_M^{-1}(d^{f_M}(\cdot,\cdot)))$ is computed instead of $f'(d(\cdot,\cdot))$. However, only distances at the pre-leaf and leaf level are dynamically modified by f_M^{-1} and f' (Fig. 5.8). The upper levels are not modified because the NM-tree stores not only direct distances between two objects (the to-parent distances) but also radii, which consist of aggregations.

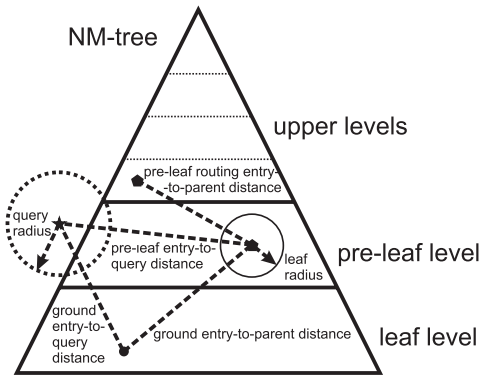


Figure 5.8: Dynamically modified distances in NM-tree

Construction

The construction of NM-tree is, in fact, the same as the construction of M-tree with the difference that T-bases for different θ must be determined from a sample of objects by TriGen before an object can be inserted into the NM-tree (Alg. 13). For this purpose, objects are stored in a sequential file until a sufficient number of objects is obtained. Then the TriGen is processed, T-bases are determined and objects from the sequential file are seriously inserted into the NM-tree using the original M-tree insertion algorithm (Alg. 7) under d^{f_M}.

Algorithm 13: NM-tree – construction

```
1  Function InsertObjectIntoNMTree(o_new)
2  Input:
3  Inserted object o_new.
4  Output:
5  None.
6  begin
7      if database size ≤ smallDBlimit then
8          store o_new into a sequential file;
       else
9          insert o_new into the NM-tree using the original M-tree insertion
           algorithm under d^{f_M};
10     if database size = smallDBlimit then
11         run the TriGen algorithm on the database, having θ_M = 0 and
           θ_1, θ_2, ..., θ_k > 0 and obtaining T-bases f_M, f_{e_1}, f_{e_2}, ..., f_{e_k} with
           weights w_M, w_1, w_2, ..., w_k;
12         for each o_i in the sequential file do
13             insert o_i into the NM-tree using the original M-tree insertion
               algorithm under d^{f_M};
14         empty the sequential file;
```

Range and kNN Queries

The range query on the NM-tree is similar to the range query on the M-tree with the difference that each pruning condition must be correctly changed. The conditions are also a bit different for non-leaf and pre-leaf nodes. A modification of the function *RangeQueryIter* for the NM-tree is shown in Alg. 14. The kNN query can be modified the same way, see [152] for details.

Algorithm 14: NM-tree – range query

```
1  Function RangeQueryIter(N, q, r, S, e_k)
2  Input:
3  Node N, query object q, radius r, set of objects S, retrieval error e_k.
4  Output:
5  Set of objects S.
6  begin
7      let o_p be the parent object of node N;
8      if N is not a leaf node then
9          if N is at pre-leaf level then
10             for each rout(o_i) ∈ N do
11                 if
                   |f_{e_k}(d(o_p, q)) − f_{e_k}(f_M^{-1}(d^{f_M}(o_i, o_p)))| ≤ f_{e_k}(r) + f_{e_k}(f_M^{-1}(r_{o_i}^{f_M}))
                   then
12                     compute d(o_i, q);
13                     if f_{e_k}(d(o_i, q)) ≤ f_{e_k}(r) + f_{e_k}(f_M^{-1}(r_{o_i}^{f_M})) then
14                         RangeQueryIter(ptr(T(o_i)), q, r, S, e_k);
15         else
16             for each rout(o_i) ∈ N do
17                 if |f_M(d(o_p, q)) − d^{f_M}(o_i, o_p)| ≤ f_M(r) + r_{o_i}^{f_M} then
18                     compute d(o_i, q);
19                     if f_M(d(o_i, q)) ≤ f_M(r) + r_{o_i}^{f_M} then
                           RangeQueryIter(ptr(T(o_i)), q, r, S, e_k);
20     else
21         for each grnd(o_i) ∈ N do
22             if |f_{e_k}(d(o_p, q)) − f_{e_k}(f_M^{-1}(d^{f_M}(o_i, o_p)))| ≤ f_{e_k}(r) then
23                 compute d(o_i, q);
24                 if d(o_i, q) ≤ r then
                       append o_i into S;
25     return S;
```

Chapter 6

Non-metric Similarity Search in Mass Spectra Databases

In this chapter, an approach for identification of peptide sequences from HPLC-MS/MS spectra based on fast and approximate non-metric similarity search in a database of theoretical spectra generated from a database of known protein sequences is proposed [153] [154]. We describe the mass spectra similarity functions utilized by MAMs, the method how we speed up the identification of peptide sequences and the possibility how to deal with modifications in query mass spectra. Finally, an improvement is proposed which speeds up the search by clustering of query mass spectra and which increases the number of identified peptide sequences by a sequential scan of protein sequence candidates [155]. The approach has been successfully tested on query sets containing small mixtures of purified proteins (Sec. 7.7). But the utilization of MAMs on complex mixtures containing thousands of proteins is a non-trivial task. However, when the precursor mass filter is utilized instead of MAMs and when it is followed by a ranking of spectra by a modification of parameterized Hausdorff distance (originally designed for MAMs), we outperform several state-of-the-art tools on complex mixtures of proteins in both – the speed of search and the number of identified peptides (Sec. 7.8).

6.1 Similarity Functions for MAMs

When the similarity search in a database of theoretical spectra is used for the identification of peptide sequences, a pair-wise mass spectra similarity (or distance) function is a crucial component of each search engine. Below, the distances used by MAMs for fast and approximate search are defined.

6.1.1 Angle Distance

Using variants of cosine similarity (Eq. 5.9) and representation of mass spectra as high-dimensional boolean vectors (Fig. 6.1) is a common idea in mass spectrometry literature [114][119][156][53][33][157]. Here, the fuzzy cosine distance d_{fuzzy} (Eq. 4.5) is redefined in a simple notation as the angle distance d_A (Eq. 6.3).

In the high-dimensional representation of mass spectra (Fig. 6.1), the range of $\frac{m}{z}$ values in a spectrum is split into subintervals. A width of a subinterval is deter-

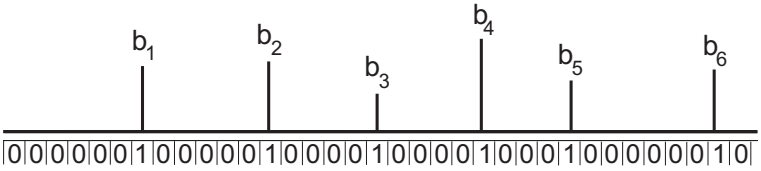


Figure 6.1: High-dimensional boolean representation of a mass spectrum

mined by $\frac{m}{z}$ error tolerance ξ (e.g., $\xi = 0.5\,\mathrm{Da}$). When a peak falls into a subinterval, a boolean vector contains 1 at the position corresponding to the subinterval, otherwise it contains 0. Instead of storing high-dimensional sparse boolean vectors, we use directly the vectors of $\frac{m}{z}$ values $\vec{x}$ and $\vec{y}$ (say, a low-dimensional representation of vectors). Considering the low-dimensional representation, two $\frac{m}{z}$ values between compared spectra are matched when $d_a(\vec{x_i}, \vec{y_j}) \leq \xi$. When the $\frac{m}{z}$ values are matched, the 1 is added to a sum. The max is used to prevent duplicate matches of the same $\frac{m}{z}$ value in one spectrum with more $\frac{m}{z}$ values in the other spectrum, i.e., every match of an $\frac{m}{z}$ value is counted only once. $dim(\vec{x})$ is the dimension of $\vec{x}$. Note that subintervals are not bounded as shown in Fig. 6.1 because the differences between $\frac{m}{z}$ values are computed.

$$d_a(\vec{x}_i, \vec{y}_j) = \begin{cases} 0, & \text{if } |\vec{x}_i - \vec{y}_j| > \xi \\ 1, & \text{else} \end{cases} \tag{6.1}$$

$$a(\vec{x}, \vec{y}) = \sum_{x_i \in \vec{x}} \max_{y_j \in \vec{y}} \{d_a(\vec{x}_i, \vec{y}_j)\} \tag{6.2}$$

$$d_A(\vec{x}, \vec{y}) = \arccos\left(\frac{a(\vec{x}, \vec{y})}{\sqrt{dim(\vec{x})dim(\vec{y})}}\right) \tag{6.3}$$

6.1.2 Logarithmic Distance

The logarithmic distance d_L (Eq. 6.5) has been proposed in [158]. The d_L is computed between short vectors of $\frac{m}{z}$ values $\vec{x}$ and $\vec{y}$ of the same dimension dim (say, $dim \in \langle 3, 8 \rangle$). For this purpose, the vectors of $\frac{m}{z}$ values are split by a sliding window to many shorter vectors of a constant size (Fig. 6.2). For example a sorted vector of 12 $\frac{m}{z}$ values is generated from the sequence *PEPTIDE*. These values correspond to y and b-ions (Fig. 2.6). The $(l-1)*2-dim+1 = 10$ vectors are created for one peptide sequence of length $l = 7$ and for vectors having the dimension $dim = 3$.

$$d_l\left(\vec{x}_i, \vec{y}_i\right) = \begin{cases} \log|\vec{x}_i - \vec{y}_i|, & \text{if } |\vec{x}_i - \vec{y}_i| > 1 \\ 0, & \text{else} \end{cases} \tag{6.4}$$

$$d_L\left(\vec{x}, \vec{y}\right) = \sum_i d_l(\vec{x}_i, \vec{y}_i) \tag{6.5}$$

The motivation for the definition of d_L is that the logarithmic distance is more robust than the Euclidean distance L_2 (Eq. 5.7) when short vectors of $\frac{m}{z}$ values of a constant size are used. A distance between mass spectra is robust if two vectors $\vec{x}$ and $\vec{y}$ are closer if there are great differences in a small number of their components

than if there are small differences in a large number of their components. For example, let's assume vectors $\vec{x} = \langle 200, 300, 400, 500 \rangle$, $\vec{y_1} = \langle 200, 300, 460, 500 \rangle$ and $\vec{y_2} = \langle 210, 305, 420, 475 \rangle$. The missing number 400 in $\vec{y_1}$ with respect to $\vec{x}$ means that the corresponding peak in the mass spectrum is missing. The superfluous number 460 in $\vec{y_1}$ refers to a noise peak, so the vectors $\vec{x}$ and $\vec{y_1}$ should be closer than $\vec{x}$ and $\vec{y_2}$. However, the L_2 distance between the vectors $\vec{x}$ and $\vec{y_1}$ is higher. The d_L distance is lower and thus it models the similarity among mass spectra better ($d_L(\vec{x}, \vec{y_1}) \doteq 1.8$, $d_L(\vec{x}, \vec{y_2}) \doteq 4.4$, $L_2(\vec{x}, \vec{y_1}) = 60$, $L_2(\vec{x}, \vec{y_2}) \doteq 33.9$).

In the approach for identification of peptide sequences from mass spectra proposed in [158], the short vectors of a constant size are used instead of vectors of $\frac{m}{z}$ values in order to increase the number of identified peptide sequences. To speed-up the search, the database of short vectors is indexed by M-tree. The speed up is about 1000× w.r.t. the sequential scan of the database of short vectors generated from theoretical spectra. However, the number of correctly assigned peptide sequences to the mass spectra by d_L is only 48%.

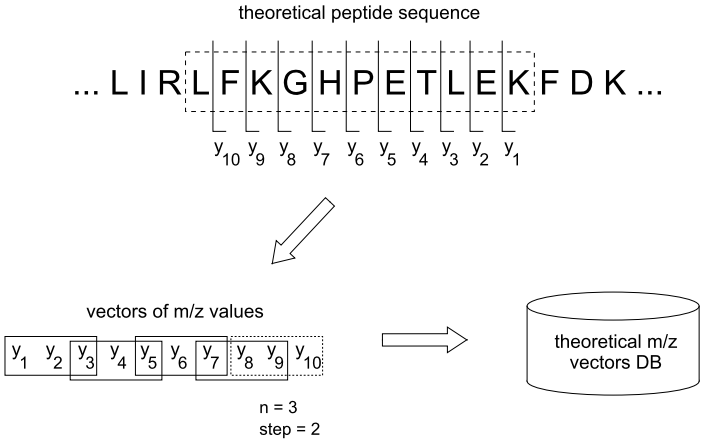


Figure 6.2: Generation of vectors of a constant size from vectors of $\frac{m}{z}$ values

6.1.3 Parameterized Hausdorff Distance

An advantage of the Hausdorff distance d_H (Eq. 5.12) is that components on different positions in two vectors can be compared. Another advantage is that d_H allows a comparison of vectors of different sizes what is a valuable property for peptide sequence identification because the mass spectra (theoretical or experimentally obtained) have usually different numbers of peaks. The advantage of logarithmic distance d_L (Eq. 6.5) is that two vectors $\vec{x}$ and $\vec{y}$ are closer considering peptide identification if there are great differences in a small number of their components than if there are small differences in a large number of their components.

The parameterized Hausdorff distance d_{HP} (Eq. 6.8) [159] combines the advantages of d_H and d_L while the logarithm is replaced by the n^{th}-root to increase the number of identified peptide sequences. The d_{HP} is a semi-metric and it

Algorithm 15: Parameterized Hausdorff Distance

```
1  Function ComputeAsymmetric(x, y, ξ, n)
2  Input:
   Sorted vectors of m/z values x and y;
   precursor mass error tolerance ξ;
   index of the root n.
3  Output:
   h(x⃗, y⃗) (Eq. 6.7).
4  begin
5      sum = 0;
6      mem_j = 0;
7      for i = 0 to x.size() − 1 do
8          /* j in the current iteration is ≥ j in the previous
              iteration (mem_j) because x and y are sorted        */
9          min = |x[i] − y[mem_j]|;
10         for j = mem_j + 1 to y.size() − 1 do
11             if |x[i] − y[j]| < min then
12             min = |x[i] − y[j]|;
13             mem_j = j;
14             /* a better result cannot be found                 */
15             else break;
16         if min > ξ then sum += ∛(min − ξ);
17     return sum / x.size();

18 Function Compute(x, y, ξ, n)
19 Input:
   Sorted vectors of m/z values x and y;
   precursor mass error tolerance ξ;
   index of the root n.
20 Output:
   d_HP between vectors x and y (Eq. 6.8).
21 begin
22     left = ComputeAsymmetric(x, y, ξ, n);
23     right = ComputeAsymmetric(y, x, ξ, n);
24     if left > right then return left;
25     return right;
```

works as follows (Fig. 6.3). First, a $\frac{m}{z}$ value in the minimum distance in the vector/spectrum $\vec{y}$ is found for each peak in $\vec{x}$. Then the n^{th}-root is applied on each of the minimum distances and a sum of roots is computed. The n^{th}-root causes that pairs of noise peaks in small distances (exceeding a small error tolerance ξ) have big contributions in the sum and that pairs of noise peaks in big distances have small contributions in the sum (in order to decrease their impact on the sum). Since numbers of peaks in compared spectra may be different, an average is computed. The process is repeated with vectors $\vec{x}$ and $\vec{y}$ switched and the maximum value is selected to obtain a symmetric measure. The $dim(x)$ is the dimension of $\vec{x}$.

$$d_h(\vec{x}_i, \vec{y}_j) = \begin{cases} |\vec{x}_i - \vec{y}_j|, & \text{if } |\vec{x}_i - \vec{y}_j| > \xi \\ 0, & \text{else} \end{cases} \tag{6.6}$$

$$h(\vec{x}, \vec{y}) = \frac{\sum_{\vec{x}_i \in \vec{x}} \sqrt[n]{\min_{\vec{y}_j \in \vec{y}} \{d_h(\vec{x}_i, \vec{y}_j)\}}}{dim(\vec{x})} \tag{6.7}$$

$$d_{HP}(\vec{x}, \vec{y}) = \max(h(\vec{x}, \vec{y}), h(\vec{y}, \vec{x})) \tag{6.8}$$

Another improvement is a significant reduction of the number of vectors that are generated from protein sequences. Only one vector of $\frac{m}{z}$ values represents a peptide sequence (theoretical spectrum) what makes this method more practically usable. This is not possible when the d_L is utilized because the splitting of vectors of $\frac{m}{z}$ values is required to achieve a sufficient number of identified peptide sequences.

The d_{HP} outperforms d_A in the number of identified peptides and in the indexability by MAMs. The number of identified peptide sequences by d_{HP} increases with increasing n (Sec. 7.4.1 and Sec. 7.8.3). For $n \to \infty$, the n^{th}-root converges to 1. A drawback is that the intrinsic dimensionality ρ also increases with increasing n (Sec. 7.4.2).

Since $\vec{x}$ and $\vec{y}$ are implicitly sorted, the d_{HP} can be computed in linear time complexity $O(dim(\vec{x}) + dim(\vec{y}))$ unlike the general Hausdorff distance d_H [116]. The asymmetric part of d_{HP} can be computed using two nested loops (Alg. 15, line 1). The inner loop (line 10) is broken when the minimum distance between

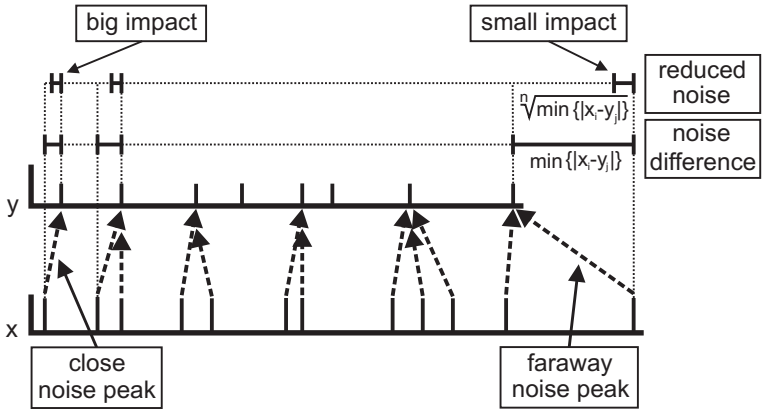


Figure 6.3: The principle of d_{HP} (the dashed arrows indicates the closest peaks in $\vec{y}$ to the peaks in $\vec{x}$).

$\frac{m}{z}$ values in the vectors $\vec{x}$ and $\vec{y}$ is found. The position of the minimum is stored in mem_j and is used as a starting value of the inner cycle in the next outer cycle. The time-consuming computation of the n^{th}-root function does not cause any problem because the range of masses corresponding to generated peptide sequences is limited and thus a table of all possible roots having limited precision up to several decimal places can be precomputed to speed up the search.

6.1.4 Modification of Parameterized Hausdorff Distance

A semi-metric modification of d_{HP} called d_{HP}^{match} (Eq. 6.10) is also proposed which increases the number of identified peptides [160]. In contrast to d_{HP}, the sum of $\frac{m}{z}$ ratios in d_{HP}^{match} is divided by the number of matches of peaks in a theoretical spectrum with peaks in a query spectrum, i.e., $a(\vec{x}, \vec{y})$ (Eq. 6.2). The 1 is added to $a(\vec{x}, \vec{y})$ to prevent the division by zero when $a(\vec{x}, \vec{y}) = 0$.

$$h^{match}(\vec{x}, \vec{y}) = \frac{\sum_{\vec{x}_i \in \vec{x}} \sqrt[n]{\min_{\vec{y}_j \in \vec{y}} \{d_h(\vec{x}_i, \vec{y}_j)\}}}{dim(\vec{x})(a(\vec{x}, \vec{y}) + 1)} \tag{6.9}$$

$$d_{HP}^{match}(\vec{x}, \vec{y}) = \max(h^{match}(\vec{x}, \vec{y}), h^{match}(\vec{y}, \vec{x})) \tag{6.10}$$

6.1.5 Angle Distance with Precursor

Since d_A has high intrinsic dimensionality ρ, the tandem cosine distance d_{tandem} is defined in Eq. 4.7 to decrease the ρ by combining d_{fuzzy} with $d_{precursor}$ (Eq. 4.6). Here, d_{tandem} is re-defined as d'_A (Eq. 6.11) which utilizes d_A. Let $s_x = \{\vec{x}, M_x\}$ be a mass spectrum where $\vec{x}$ is the vector of $\frac{m}{z}$ values and M_x is the precursor mass of the spectrum s_x. The higher the c_2, the lower the ρ. For $c_1 = 0$, the distance corresponds to the precursor mass filter. For $c_2 = 0$, the distance corresponds to d_A.

$$d'_A(s_x, s_y) = c_1 d_A(\vec{x}, \vec{y}) + c_2 d_{precursor}(M_x, M_y) \tag{6.11}$$

6.1.6 Parameterized Hausdorff Distance with Precursor

The d_{HP} can be combined with $d_{precursor}$ as well as the d'_A (Eq. 6.12). In fact, this approach can be applied to any distance function to reduce the high intrinsic dimensionality ρ.

$$d'_{HP}(s_x, s_y) = c_1 d_{HP}(\vec{x}, \vec{y}) + c_2 d_{precursor}(M_x, M_y) \tag{6.12}$$

6.2 Identification of Peptide Sequences

In this section, the method for fast and approximative identification of peptide sequences from HPLC-MS/MS spectra is proposed, where (non)metric access methods are utilized as database indexing techniques to speed up the search in databases of theoretical mass spectra [154] [153]. The entire process of peptide sequences identification, incorporating the previously defined measures (Sec. 6.1), can be split into two phases – indexing the database of theoretical mass spectra

and querying the database with a set of query spectra. A simple identification workflow is shown in Fig. 6.4. First, an index is created to speed-up the search in the database of theoretical mass spectra generated from a database of known protein sequences. Second, the index is queried with the set of experimental (or query) spectra while peptide sequences are being identified.

6.2.1 Indexing

The indexing phase can be formalized as follows:

1. Protein sequences in the database are split to peptide sequences "in silico". The splitting rules are determined by the digestion enzyme (Tab. 2.1). A commonly used enzyme is the trypsin which splits the protein chains after each amino acid lysine (K) and arginine (R) if they are not followed by proline (P) [9]. However, even though the splitting sites are well predictable, the process is not perfect in practice and some missed cleavage sites can occur. Thus the *maximum number of missed cleavage sites* is adjusted as a parameter what is a common option in any search engine for peptide sequences identification.

2. The theoretical mass spectra are generated from the peptide sequences. The $\frac{m}{z}$ values of fragment ions commonly occurring in the experimental spectra are generated from each peptide sequence, e.g., y, b-ions or y, b, y^{2+}-ions (Tab. 2.2). A vector of $\frac{m}{z}$ values formed from the peptide sequence of a length l has the dimension $n(l-1)$ where n is the number of generated fragment ion series, e.g., $n = 2$ when y-ions and b-ions are generated in a theoretical spectrum (Fig. 2.6). Each theoretical spectrum corresponds to one indexed vector. The vectors of $\frac{m}{z}$ values are sorted in ascending order.

3. The vectors of $\frac{m}{z}$ values are indexed by a MAM (e.g., by M-tree or LAESA) under a given distance (e.g., d_{HP} or d_A) modified by TriGen (Sec. 5.2.4). Note that any non-metric function (e.g., SEQUEST-like scoring (Sec. 3.2.1)) might be turned into a metric by the TriGen algorithm. A drawback is that its efficiency is not guaranteed and likely the intrinsic dimensionality ρ will

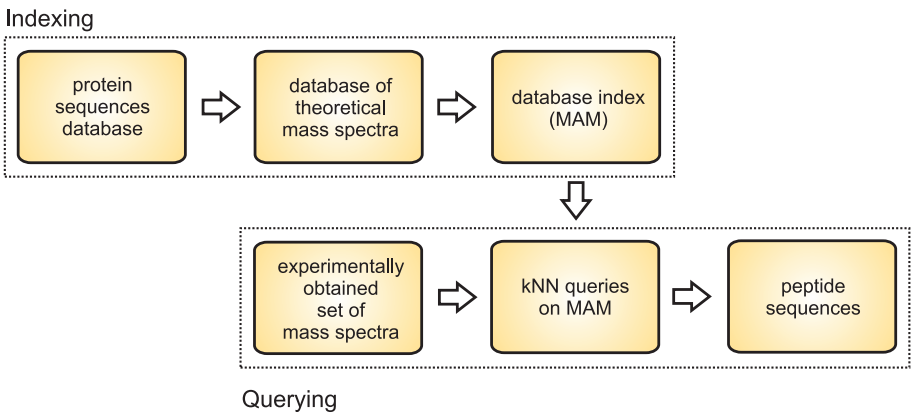


Figure 6.4: Workflow of peptide sequences identification by MAMs

be high. Since the d_{HP} is almost the metric ($\theta \doteq 0$) the TriGen is not necessary in the indexing phase.

To decrease the main memory requirements and to ensure the constant size of items indexed by a MAM, an indexed object does not store a vector of $\frac{m}{z}$ values but it stores two pointers into a database of protein sequences which is entire loaded in the main memory. The pointers refer to the begin and to the end of a peptide sequence. The sorted vector of $\frac{m}{z}$ values is generated from the peptide sequence when the d_{HP} is called. This approach slows down the search up to 25% – 33% but the savings of the main memory are essential when large databases of protein sequences are being processed. The approach can be further optimized when the length of a peptide sequence is stored instead of the pointer to the end of the peptide sequence.

6.2.2 Querying

During the indexing phase, a MAM is created which indexes a virtual database of theoretical mass spectra generated from a database of protein sequences. Peptide sequences corresponding to the query mass spectra are determined by querying the MAM as follows:

1. In order to increase both the efficiency and effectiveness, the query spectra are preprocessed by a heuristic before the MAM is queried. Simple heuristics are described in Sec. 3.1.1. The peak selection heuristic impacts both – the effectiveness and efficiency of the search. The effectiveness is impacted by the way how the peaks are being selected. A good heuristic should select such peaks from query spectra which correspond to the peaks generated in theoretical spectra. The efficiency of the search is impacted by the number of selected peaks because the intrinsic dimensionality ρ increases with the number of peaks.

2. When the query spectra contain modifications, an additional pre-processing must be performed (Sec. 6.3). During the pre-processing, shifts of $\frac{m}{z}$ values corresponding to the modifications are generated into the query spectra. A disadvantage is that a selection of a high number of modifications to be supported may lower the number of identified peptide sequences and slow down the search because the intrinsic dimensionality ρ increases with the number of peaks in a query spectrum.

3. A kNN query is performed on the MAM for each query spectrum, thus k nearest peptide sequences (theoretical mass spectra, respectively) are selected for each query spectrum.

4. Even though the correct peptide sequence corresponding to the spectrum is obtained as the nearest neighbor ($k = 1$) in many cases, the set of returned peptide sequences (when $k > 1$) can be re-ranked using a more sophisticated similarity function, for which satisfying of the metric postulates and low ρ are not required (e.g., SEQUEST-like scoring (Sec. 3.2.1)).

We assume that a query spectrum is successfully interpreted when the correct peptide sequence is among the k nearest neighbors (regardless of its position

in the kNN result set). An additional similarity function is assumed in a real-world application, which determines the correct peptide sequence from the set of k candidate peptide sequences. Such a refining similarity function can consider other fragment ions than y-ions and b-ions (Tab. 2.2). Since the other fragment ions occur rarely in the mass spectra, they can help to determine the correct peptide sequence from the k candidate sequences. On the other hand, these fragment ions may worsen the effectiveness and efficiency when the MAM is queried because they occur rarely. Thus it is not very suitable to generate $\frac{m}{z}$ values of all possible fragment ion series into theoretical spectra.

6.3 Dealing with Modifications in Spectra

Due to the complexity of the similarity search of query mass spectra with modifications, this problem is usually neglected in existing indexing approaches (Chap. 4). Here, the approach based on d_{HP} is extended to support the processing of modified spectra [153] [161]. This extension could be also employed in other approaches for peptide sequences identification from query spectra with modifications.

When a query spectrum contains modifications, some peaks in the spectrum are shifted. The shifts depend on the positions of modifications occurring in the peptide, i.e., which amino acids in the sequence have modified masses (Fig. 6.5). When the modifications occurring in the query spectra are known before the search (commonly they are defined by the user), two basic ways can be used to support them. First, peaks in theoretical spectra can be shifted for any defined modification or any combination of modifications. The theoretical spectra can be indexed by the MAM while the query is unchanged. A drawback is that the size of the database of theoretical spectra grows exponentially when many modifications or combinations of modifications are defined. The second way is to change the query spectrum while the database remains unchanged. For our purposes, the latter approach is employed. The entire process of query construction for one modification α can be summarized as follows:

1. Let S_T be the theoretical spectrum of a peptide sequence (Fig. 6.6a). Let $S_0 = \langle m_1, ..., m_p \rangle$ (Fig. 6.6b) be a query spectrum having p peaks ($\frac{m}{z}$ ratios where $z = 1$).

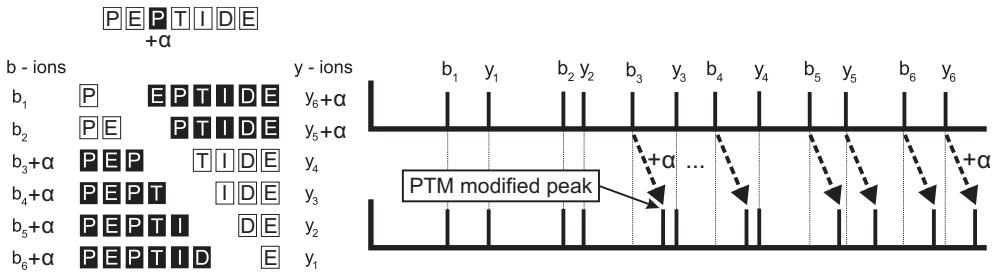


Figure 6.5: Peptide with a modification α (black peptide fragments are affected by the modification – their masses are modified and corresponding peaks are shifted)

2. When a modification α (e.g., $\alpha = 57$) happens at an unknown position i in the peptide, only m_i and some of the following peaks are shifted. Since we cannot predict this position, the entire spectrum is shifted by $-\alpha$. A shifted vector of the spectrum S_0 for the modification α is $S_\alpha = \langle m_1 - \alpha, ..., m_p - \alpha \rangle$ (Fig. 6.6c). Thus peaks shifted by α in S_0 have their "unshifted" counterparts in S_α.

3. S_0 and S_α are appended (Fig. 6.6d), where the union of spectra $S_0 \cup S_\alpha$ is a sorted vector of all peaks in the spectra S_0 and S_α.

4. While S_0 forms the query for an unmodified spectrum, the query for the spectrum with the modification α is $S_I = S_0 \cup S_\alpha$.

A disadvantage is that two other types of noise peaks occur in queries. First, the peaks shifted "in vitro" in S_0 which are superfluous in the union $S_0 \cup S_\alpha$. Second, the artificial noise peaks computed in S_α, which were not modified and they should not have been shifted in S_α. These two types of noise peaks cannot be removed, because we are not able to recognize them. Since mass spectra contain many noise peaks and d_{HP} is able to reduce them, the other noise peaks are partially reduced as well.

In case of two modifications α and β, the query is represented by spectrum $S_{II} = S_0 \cup S_\alpha \cup S_\beta \cup S_{\alpha+\beta}$, where $\alpha + \beta$ are peaks shifted by both modifications at once. In case of three modifications α, β and γ, the query is represented by spectrum $S_{III} = S_0 \cup S_\alpha \cup S_\beta \cup S_\gamma \cup S_{\alpha+\beta} \cup S_{\alpha+\gamma} \cup S_{\beta+\gamma} \cup S_{\alpha+\beta+\gamma}$, etc. Since lengths of peptide sequences are limited, the number of modifications per spectrum usually does not exceed 2 or 3 (Tab. 7.1). Therefore the maximum number of shifted spectra unified in the query, which might be up to $\sum_{i=0}^{n} \binom{n}{i}$ for n different modifications, is not reached in practice.

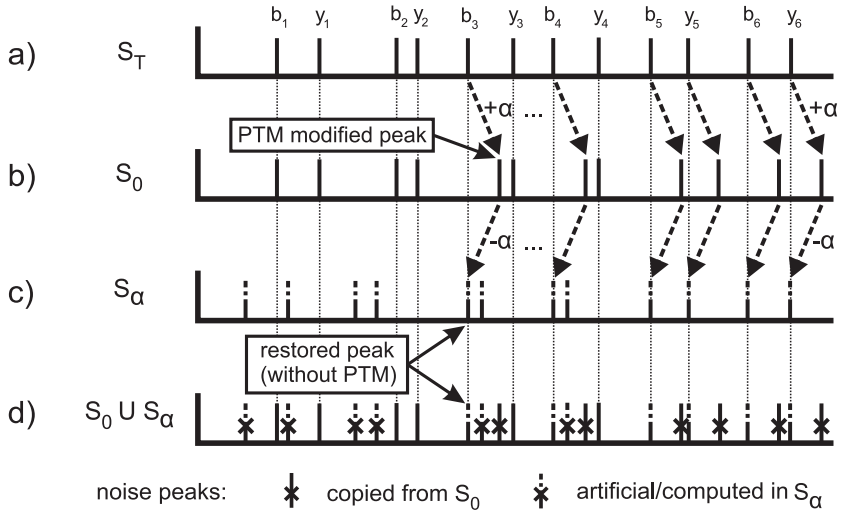


Figure 6.6: Dealing with modifications (S_T corresponds to S_0 with the modification α happened at position 3 in the respective peptide sequence)

6.4 Clustering and Sequential Scan of Protein Sequence Candidates

Commonly, an "in vitro" protein sample is analyzed by more spectrometer runs. A set containing up to tens of thousands of mass spectra is captured in each run. The proteins in the sample are split to many peptide ions where a mass spectrum corresponds to a peptide ion. More peptide ions correspond to a peptide sequence and, similarly, more peptide sequences come from a protein sequence.

The original approach for identification of peptides is fast but approximative, because $\theta > 0$ is used to lower the ρ (to speed-up the search, respectively). Since a query set of mass spectra contains multiple spectra which fall into a single protein sequence, the effectiveness of the search can be improved by a sequential scan of protein sequence candidates. The sequential scan reveals PSMs missed because of the approximate search (Fig. 6.7c). Since the query set of spectra contains more spectra for the same peptide sequence (i.e., sibling spectra), the efficiency of the search can be improved by clustering when the sibling spectra are being detected and removed from the query set (Fig. 6.7a) [155].

While the clustering is employed in a pre-processing step to filter out the noise spectra and thus to speed up the search, the sequential scan of the candidate protein sequences is used in a post-processing step to increase the number of identified peptide sequences. The improved workflow for peptide/protein sequences identification is shown in Fig. 6.7. Note that the improvement is suitable particularly for small mixtures of proteins. When a complex sample is analyzed (i.e. thousands of proteins), the sequential scan of protein sequence candidates can be time-consuming. Below, the pre-processing, the query phase and the post-processing are described in detail.

6.4.1 Pre-processing

Despite the search in an indexed database is fast, query sets still contain many noise spectra which can be ignored. The noise spectra cannot be assigned to peptide sequences because they occur as an artifact of the spectrometer process. A preprocessing can be used to eliminate the noise spectra and to speed up the search, because only a small part of the query set needs to be interpreted.

The pre-processing can be realized by clustering of siblings spectra (Fig. 6.7a).

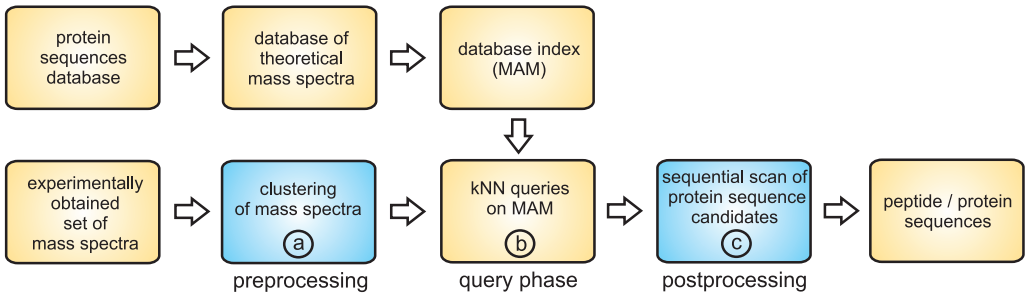


Figure 6.7: Improved workflow for peptide/protein sequences identification (original method is yellow, improvements are blue)

Even though a set of query spectra commonly contains more spectra corresponding to a peptide sequence, it is advantageous when query sets from more spectrometer runs can be appended. Hence, many interpretable spectra which are captured only once per a spectrometer run have a sibling spectrum in the query set and they are not eliminated by the clustering. On the other hand, the noise spectra are successfully cleared away and thus many spectra do not have to be searched in the query phase (Fig. 6.7b), making the search significantly faster.

We use a simple hierarchical clustering based on the complete linkage (Alg. 16), i.e., all spectra in a cluster are similar to each other. The algorithm requires a set of clusters C initialized with one mass spectrum per cluster. Then two phases are repeated in w cycles. First, pairs of clusters in the minimum $d_{HP}(c_{i,0}, c_{j,0})$ such that $d_{HP}(c_{i,0}, c_{j,0}) \leq t$ are merged, where t is a threshold of the d_{HP} and $c_{i,0}$, $c_{j,0}$ are the centroids of clusters c_i, c_j (Alg. 17). The threshold t determines whether the spectra in a cluster are similar or not. Moreover, it determines the number of clusters. If t is too low, each spectrum forms a singleton (a cluster containing one spectrum). If t is too high, all spectra form one big cluster.

Second, the spectra are rearranged among the clusters (Alg. 19). A spectrum is moved to another cluster, if the d_{HP} between the query spectrum and any spectrum in the target cluster is less or equal t and the difference of precursor masses is less or equal λ. In case that more clusters are selected, the cluster is picked where the d_{HP} between its centroid and the moved object is minimal.

New centroids of clusters are selected after each phase (Alg. 18). Finally, the centroids of clusters containing at least two spectra form the queries, which are processed by the MAM. Another way consists in putting all peaks from all the spectra in the cluster into a representative spectrum. The intensities of the closest peaks are counted up and their $\frac{m}{z}$ values are averaged. Since the increasing number of peaks in a spectrum worsens the efficiency of MAMs because of high intrinsic dimensionality ρ, this alternative needs to be a bit of improved. For example, a specified number of peaks with the highest intensity might be selected from the representative spectrum.

6.4.2 Query phase

The query phase (Fig. 6.7b) corresponds to the original idea proposed in Sec. 6.2.2, where a kNN query is performed by a MAM for each query spectrum selected in the pre-processing step. The k nearest neighbor peptide sequences to each query spectrum are assigned to the protein sequences of their origin. The protein sequences containing at least one "good" peptide sequence hit (e.g., $d_{HP} \leq 0.65$) form the protein sequence candidates.

6.4.3 Post-processing

The post-processing is a sequential scan of protein sequence candidates (Fig. 6.7c), which significantly improves the number of identified peptide sequences because more peptide sequences in a protein sequence correspond to the mass spectra in the query set [162] [23]. The protein sequence candidates (i.e., a small subset of the database sequences selected in the query phase) are split to peptide sequences and their theoretical spectra are compared to the entire set of input query spectra

(as it was before the clustering phase). Many spectra previously missed during the pre-processing or during the query phase are assigned to peptide sequences. The newly identified peptide sequences are assigned to the protein sequences of their origin. Finally, the peptide (or protein, respectively) sequences identified in the query phase and refined in the post-processing phase form the result. Note that some peptide sequences are lost during the clustering because their spectra are present only once in the query set. Some peptide sequences are lost during the query phase because the search is only approximative (non-metric). The sequential scan of protein sequence candidates helps to reveal a peptide sequence in case it forms a part of a candidate protein sequence which is hit by another peptide sequence.

Algorithm 16: Clustering of query mass spectra (main function)

```
1  Function Clustering(C, t, w)
2  Input:
     A set of clusters C initialized with one query mass spectrum per cluster;
     a threshold t;
     a number of clustering cycles w.
3  Output:
     The set of clusters C.
4  begin
5      for i = 1 to w do
6          MergeClusters(C, t);
7          SelectCentroids(C);
8          RearrangeClusters(C, t);
9          SelectCentroids(C);
```

Algorithm 17: Clustering of query mass spectra (*MergeClusters*)

```
1  Function MergeClusters(C, t)
2  Input:
     A set of clusters C;
     a threshold t.
3  Output:
     The set of clusters C.
4  begin
5      for all clusters c_i ∈ C do
6          for all clusters c_j ∈ C do
7              if i ≠ j then
8                  /* a centroid of the cluster c_i is stored at c_{i,0}  */
9                  if all spectra c_{j,k} in the cluster c_j have
                   d_HP(c_{i,0}, c_{j,k}) ≤ t
                   then
10                     store the position j of the cluster c_j in the minimum
                       d_HP(c_{i,0}, c_{j,0}) into p;
11         merge the clusters c_i and c_p;
```

Algorithm 18: Clustering of query mass spectra (*SelectCentroids*)

```
1  Function SelectCentroids(C)
2  Input:
     A set of clusters C.
3  Output:
     The set of clusters C.
4  begin
5      for all clusters c_i ∈ C do
6          P = ∅;
7          for all spectra c_{i,j} ∈ c_i do
8              for all spectra c_{i,k} ∈ c_i do
9                  if j ≠ k then
10                     store the maximum distance d_HP(c_{i,j}, c_{i,k}) and the
                       position k of spectrum c_{i,k} in the maximum d_HP into P;
11         select the position p in the minimum d_HP from P;
12         /* a new centroid is being moved to c_{i,0}                    */
13         switch the spectra c_{i,0} and c_{i,p};
```

Algorithm 19: Clustering of query mass spectra (*RearrangeClusters*)

```
1  Function RearrangeClusters(C, t)
2  Input:
     A set of clusters C;
     a threshold t.
3  Output:
     The set of clusters C.
4  begin
5      for all clusters c_i ∈ C do
6          for all spectra c_{i,k} ∈ c_i do
7              P = ∅;
8              for all clusters c_j ∈ C do
9                  for all spectra c_{j,l} ∈ c_j do
10                     if all spectra c_{j,l} have d_HP(c_{i,k}, c_{j,l}) ≤ t then
11                         store the distance d_HP(c_{i,k}, c_{j,0}) and the position j of
                           the cluster c_j into P;
               select the position p of the cluster in the minimum d_HP from P;
               move the spectrum c_{i,k} into the cluster c_p;
```

Chapter 7

Experiments

In this chapter, an experimental evaluation of the methods described in Chap. 6 is proposed. First, the quantities measured in the experiments and the data sets are described. Then the T-bases for d_{HP} and d_A determined by the TriGen algorithm are proposed. The T-bases enable the utilization of the distances for fast and approximate search by MAMs.

The efficiency and effectiveness of peptide sequences identification by non-metric access methods are studied without and with the support of modifications [153]. The utilization of different indexing structures is analyzed (M-tree, LAESA and NM-tree) [154] [163]. An improvement of this approach is also tested where a pre-processing by clustering is employed to speed up the search and a post-processing by a sequential scan of protein sequence candidates is used to increase the number of identified peptide sequences [155].

Finally, a comparison of the state-of-the-art tools with the parameterized Hausdorff distance d_{HP} and with the modification of the parameterized Hausdorff distance d_{HP}^{match} is proposed where the precursor mass filter is utilized instead of MAMs [160].

7.1 Measured Quantities

The following quantities were measured in the experiments utilizing MAMs:

1. The *correctness of mass spectra interpretation* (correctness/quality of peptide sequences identification or ratio of identified spectra) as a ratio of mass spectra correctly annotated with peptide sequences to all spectra in the query set (i.e., to a ground truth determined by another peptide identification engine). We assume that a peptide sequence is correctly annotated when the correct peptide sequence is among the k nearest neighbors to the query spectrum.

2. The *distance computations ratio* as a ratio of the average number of distance computations (the number of calls of a pair-wise distance function, e.g., d_{HP}) per one mass spectrum interpretation to the cost of the sequential scan.

3. The *average query time* per one mass spectrum interpretation.

The correctness of interpretation can be also understood as *effectiveness* while the distance computations ratio and the average query time as *efficiency* of the search.

7.2 Data Sets

Four different setups of databases and query sets were used in the experiments. Below, the setups are briefly described.

7.2.1 Amethyst and Opal

In the first setup, a unification of query sets *Amethyst* and *Opal* of human mass spectra was used [164]. The annotations of spectra (peptide sequences corresponding to the mass spectra) were known. The query sets contained mass spectra without and with modifications (Tab. 7.1). The database of protein sequences was an extension of a list of protein sequences corresponding to the set of experimental spectra. The database was extended with protein sequences from MSDB (Mass Spectrometry Protein Sequence Database, release 08-31-2006) [2] and contained 100,000 protein sequences (5,612,211 peptide sequences).

The experiments were carried out on a machine with 2 processors Intel Xeon E5450 (8 cores × 3GHz), 8 GB RAM and 64-bit OS Windows Server 2008 R2. Following settings were used unless otherwise specified – n^{th} root in d_{HP}: 30; digestion enzyme: trypsin ([KR]/P); maximum number of missed cleavage sites: 1; ξ: 0.4 Da; number of peaks with highest intensity selected from experimental spectra: 50; fragment ions generated in theoretical mass spectra: y, b.

Query set	**Modifications per spectrum**						
	0	**1**	**2**	**3**	**4**	**5**	**6**
Amethyst	1095	371	85	13	2	2	1
Opal	239	237	51	8	1	0	0

Table 7.1: Numbers of spectra in query sets Amethyst and Opal

7.2.2 Keller 1

In the second setup, MS/MS spectra from *Keller et al.* [57] were used. The spectra were obtained from 22 mass spectrometer runs on two protein mixtures A and B where 18 proteins were mixed together. 14 mass spectrometer runs were performed on the mixture A and 8 runs on the mixture B. Two query sets were used in this setup. The first set Q_1 contained 119 spectra from the first run on mixture A and the second query set Q_2 contained 1941 spectra from all runs on both mixtures. Only the annotated spectra and spectra split by trypsin were selected. The annotations were determined by SEQUEST [55] and the results were manually checked by domain experts. Hence, we consider the SEQUEST-interpreted results as the confirmed ground truth.

The databases DB_1 and DB_2 were extensions of a file with correct protein sequences assigned to the mass spectra. The databases were extended with protein

sequences from MSDB (release 08-31-2006) [2]. The DB_1 contained 100,000 protein sequences (5,600,747 peptide sequences) and DB_2 contained 500,000 protein sequences (29,460,880 peptide sequences).

The experiments were carried out on a machine with 2 processors Intel Xeon E5450 (8 cores × 3GHz) with 8 GB RAM and 64-bit OS Windows Server 2008 R2. By default, the average query time per one mass spectrum interpretation was measured on 8 processor cores. The following settings were used unless otherwise specified – n^{th} root in d_{HP}: 50; digestion enzyme: trypsin ([KR]/P); maximum number of missed cleavage sites: 1; ξ: 0.4 Da; precursor mass error tolerance (λ): 2 Da; number of peaks with highest intensity selected from experimental spectra: 100; fragment ions generated in theoretical mass spectra: y, b.

7.2.3 Keller 2

In the third setup, MS/MS spectra from*Keller et al.* [57] were also used. The spectra from the first 6 runs on mixture A and from all runs on mixture B were used. The query spectra were searched against the database DB_1 (Sec 7.2.2).

The experiments were carried out on a machine with 2 processors Intel Xeon X5660 (24 cores, 2.8 GHz) with 24 GB RAM and 64-bit OS Windows Server 2008 R2. The stated average query times of clustering and peptide sequences identification are measured on one core. If not otherwise stated, the following settings were used – digestion enzyme: trypsin ([KR]/P); maximum number of missed cleavage sites: 1; mass range of peptide ions generated from the database: 500-5,000 Da; fragment ions generated in hypothetical mass spectra: y-ions and b-ions; mass range of generated fragment ions: 300-2,000; $\frac{m}{z}$ error tolerance (ξ): 0.4 Da; number of peaks with highest intensity used in a query: 50; distance measure: d_{HP} (with $n = 30$); clustering threshold (t): 0.65 (values returned by d_{HP} were normalized to $\langle 0, 1 \rangle$), T-error tolerance θ: 0.1.

7.2.4 E. coli and Human

HPLC-MS/MS spectra from E. coli and human were used in the last setup. Separation of the E. coli digest was performed using an easyLC HPLC system (Proxean) with a 2 h segmented gradient. Peptides eluting from the column were on-line injected into an LTQ-Orbitrap XL instrument (Thermo Fisher Scientific), with top 10 selection of the most abundant ions for further fragmentation. A dynamic exclusion list of 500 masses and exclusion time of 90 seconds was used to avoid repeated fragmentation of the same ions. The query set *E.coli* contained 30,358 tandem mass spectra. Human spectra were taken from 2 runs from a label-free human data set [165] – the query set *Hum48* contained 26,417 spectra and *Hum49* contained 24,537 spectra. The query sets are also available at [166] or [167].

The manually curated database containing 8,272 protein (332,862 peptide) sequences was used with *E.coli*. The database of 173,450 human protein (9,567,012 peptide) sequences from UniProtKB/Swiss-Prot (v. 07/2012) [3] was used with human query sets. Decoy protein sequences were included in both databases. Theoretical spectra were generated with following settings – digestion enzyme: trypsin ([KR]/P); maximum number of missed cleavage sites: 1; length of peptide sequences: 7-50 amino acids; precursor mass of peptides: 500-5,000 Da; fragment

ions types: y, b, y^{2+}; $\frac{m}{z}$ ratios of fragment ions: 200-2,000 Da. Query spectra were processed as follows – minimum number of peaks in a spectrum to be processed: 30; peak selection heuristic: the range of $\frac{m}{z}$ values was split by 50 Da, 5 most intense peaks were selected in each window and 50 most intense peaks were selected from the unification of the most intense peaks in the windows. $\lambda = 10$ ppm, $\xi = 0.5$ Da and $n = 30$ (in d_{HP} and d_{HP}^{match}). A machine with Windows 7 x64, Intel Core i7 2GHz, 8 GB RAM and 5400 rpm HDD was used.

7.3 TriGen-based Modifications

Even though the d_{HP} (Sec. 6.1.3) and d_A (Sec. 6.1.1) are semi-metric distances, the T-error of each of them is very low (below 0.001) but the intrinsic dimensionality ρ is very high ($\rho = 88.5$ for d_{HP} and $\rho = 158.1$ for d_A). Thus, the TriGen algorithm (Sec. 5.2.4) was used to improve ρ. The d_A and d_{HP} must be normalized to $\langle 0, 1 \rangle$ in order to employ the TriGen. The d_A is normalized by $\frac{\pi}{2}$, while d_{HP} is normalized by $\sqrt[n]{d_h^{max}}$, where d_h^{max} is the maximum mass of theoretically generated peptides (e.g., 5000 Da). The modifications of d_{HP} and d_A were determined for the experimental setups proposed in Sec. 7.2.1 and Sec. 7.2.2.

7.3.1 FP-bases for Amethyst and Opal

The TriGen was used on a sample of the database from Sec. 7.2.1 to improve the intrinsic dimensionality ρ, setting the T-error tolerances θ to the range 0 – 0.1. The FP-base was used. For all θ, the TriGen found convex T-modifiers ($w < 0$), so ρ was reduced (down to 2.3 for $\theta = 0.1$). The resulting FP-bases determined by TriGen for d_{HP} ($n = 30$) and d_A are shown in Tab. 7.2.

	d_{HP}		d_A	
T-error	ρ	w	ρ	w
0	88.5	-0.17	158.1	-0.84
0.01	5.2	-4.44	11.1	-7.43
0.02	4.0	-5.23	8.5	-8.94
0.03	3.5	-5.71	7.1	-10.01
0.04	3.2	-6.08	6.3	-10.92
0.05	2.9	-6.40	5.7	-11.65
0.06	2.8	-6.64	5.2	-12.34
0.07	2.6	-6.87	4.8	-13.00
0.08	2.5	-7.06	4.5	-13.63
0.09	2.4	-7.25	4.2	-14.28
0.1	2.3	-7.42	3.9	-14.92

Table 7.2: ρ and empirically determined FP-bases for d_{HP} ($n = 30$) and d_A

7.3.2 FP-bases and RBQ-bases for Keller 1

The TriGen was used on a sample of the database DB_1 from Sec. 7.2.2, setting θ to the range 0.001 – 0.2. The FP-base and 454 different RBQ-bases (different points (a, b)) were used. For all θ the TriGen found convex T-modifiers ($w < 0$), so the intrinsic dimensionality ρ was reduced (down to 2.1 for $\theta = 0.2$). The resulting T-bases with smallest ρ determined by the TriGen for d_{HP} ($n = 50$)

	FP(v,w)			$RBQ_{(a,b)}$(v,w)				
T-err.tol.	ρ	T-err.	w	ρ	T-err.	a	b	w
0.001	18.6	0.001	-2.6	15.7	0.001	0.22	0.82	-3.1
0.01	7.6	0.013	-5.0	6.8	0.013	0.22	0.82	-11.3
0.02	6.0	0.023	-5.9	5.7	0.020	0.22	0.82	-20.5
0.04	4.5	0.042	-7.0	4.6	0.042	0.13	0.83	-7.6
0.06	3.8	0.062	-7.9	3.7	0.061	0.13	0.83	-10.4
0.08	3.3	0.082	-8.6	3.1	0.081	0.13	0.83	-15.3
0.1	3.0	0.102	-9.2	2.8	0.092	0.13	0.83	-20.4
0.12	2.8	0.120	-9.6	2.3	0.112	0.13	0.83	-54.9
0.14	2.6	0.138	-10.1	2.7	0.140	0.05	0.85	-6.4
0.16	2.4	0.154	-10.5	2.5	0.160	0.05	0.85	-7.1
0.18	2.3	0.173	-10.9	2.3	0.174	0.05	0.85	-7.5
0.2	2.2	0.191	-11.3	2.1	0.196	0.05	0.85	-8.4

Table 7.3: ρ and empirically determined FP and RBQ-bases for d_{HP} ($n = 50$)

are shown in Tab. 7.3. ρ is slightly better when RBQ-bases are used than when FP-base is used, however, testing many RBQ-bases is time-consuming.

7.4 Effectiveness and Efficiency of Non-metric Similarity Search

The algorithm for identification of peptide sequences using the approximate similarity search by MAMs (Sec. 6.2) was tested on query sets Amethyst and Opal (Sec. 7.2.1) [153]. First, the correctness of interpretation using d_{HP} was tested when the sequential scan of entire database was employed. Second, the impact of T-bases determined by TriGen on the indexability by MAMs was analyzed. Finally, the speed up and correctness of interpretation on M-tree with TriGen modified d_{HP} were studied.

7.4.1 Sequential Scan

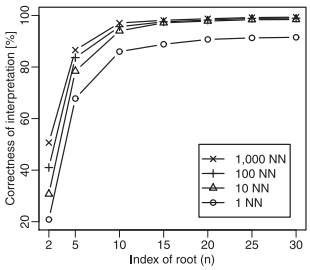


Figure 7.1: Sequential scan – correctness of interpretation (d_{HP})

The sequential scan of entire database of theoretical spectra was performed while d_{HP} was utilized. The correctness of interpretation and average query time were measured lacking spectra with modifications. The correctness of interpretation was bigger with increasing index of the root n (Fig. 7.1). It was up to 98.3% when $n = 30$ and 10 NN queries were used. The average query time was 14.4 s. The correctness of interpretation was 95.7% and the average query time was 9.8 s when d_A and 10 NN queries were used.

7.4.2 Improving the Indexability

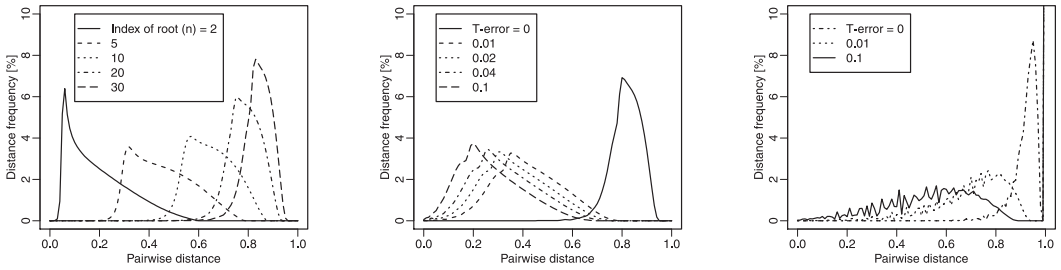


Figure 7.2: Distance distribution histograms - a) d_{HP} for increasing n, b) d_{HP} modified by TriGen, c) d_A modified by TriGen

A disadvantage of the n^{th} root function in d_{HP} is that intrinsic dimensionality ρ increases with increasing n hence the difference between MAMs and the sequential scan blends. In Fig. 7.2a see the distance distributions under d_{HP} (not modified by TriGen) for increasing n. The x-axis represents normalized pairwise distances between spectra in the database. The more the distribution is pushed to the right, the higher the intrinsic dimensionality ρ. In Fig. 7.2b,c observe the impact of the TriGen-modified d_{HP} and d_A on the distance distributions. The FP-base (Eq. 5.22) with weights proposed in Tab. 7.2 are used. The higher the T-error tolerance θ, the lower the ρ (distance distributions pushed to the left).

7.4.3 Speeding-up using M-tree

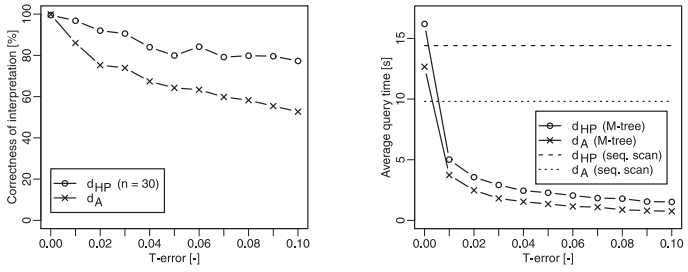


Figure 7.3: Speeding-up using M-tree – a) correctness of interpretation, b) average query time

In order to verify the behavior of d_{HP} and d_A predicted from the distance distributions, 1,000 NN queries were performed on M-trees for various θ (Fig. 7.3). On average, the correctness of interpretation was 1.3× bigger for d_{HP} than for d_A. The d_{HP} was 4.9× faster than the sequential scan while the correctness of interpretation was more than 90% ($\theta = 0.03$). The d_A was 5.4× faster than the sequential scan but the correctness was only 73.9% at the same θ. The average query time was 14.4 s for d_{HP} and 9.8 s for d_A when the sequential scan was performed.

7.5 Dealing with Modifications in Spectra

The identification of peptide sequences from query spectra with modifications has been tested on query sets Amethyst and Opal (Sec. 7.2.1). A modified query spectrum was generated for each query spectrum (Sec. 6.3).

7.5.1 Sequential Scan

Modifications per spectrum	Correctness of interpretation [%]			
	1 NN	10 NN	100 NN	1,000 NN
without the support of modifications				
1	20.0	41.0	61.7	75.0
2	9.3	18.6	28.7	65.8
3	0	0	0	0
with the support of modifications				
1	69.9	84.0	94.3	98.9
2	24.8	55.1	76.8	90.8
3	31.0	54.8	61.9	100.0

Table 7.4: Correctness of interpretation without and with the support of modifications

The correctness of interpretation was measured without and with the support of modifications (Tab. 7.4). 467 spectra containing one modification, 77 spectra containing two modifications and 10 spectra containing three modifications were used. The following modifications $\alpha \in \{-28,\ -17,\ -14,\ 1,\ 14,\ 16,\ 28,\ 57\}$, pairs of modifications $\{\alpha, \beta\} \in \{\{-17, 57\},\ \{57, 57\}\}$ and triplets of modifications $\{\alpha, \beta, \gamma\} \in \{\{-17, 57, 57\},\ \{57, 57, 57\}\}$ were searched. The queries S_I were performed for spectra with one modification, S_{II} for spectra containing two modifications and S_{III} for spectra containing three modifications (Sec. 6.3).

Since modifications commonly do not shift all peaks in the query spectra, the d_{HP} is partially able to determine correct peptide sequences when modifications in the query spectra are not supported. The correctness of interpretation is more than 90% in all cases when modifications are supported (1,000 NN queries). It decreases with increasing number of modifications per spectrum when smaller kNN queries are used.

The number of peaks in a query and the average query time increase with increasing number of supported modifications. The average query time is 18.9 s for spectra with one modification, 24.2 s for spectra with two modifications and 35.6 s for spectra with three modifications.

7.5.2 Speeding-up using M-tree

A set of 1,000 NN queries was processed on M-trees for different θ when the d_{HP} was employed. Results for spectra with one and two modifications are shown in Fig. 7.4. The M-tree was 3.3× faster for spectra with one modification and 2.5× faster for spectra with two modifications than the sequential scan (T-error = 0.06) while the correctness of interpretation was still about 90%.

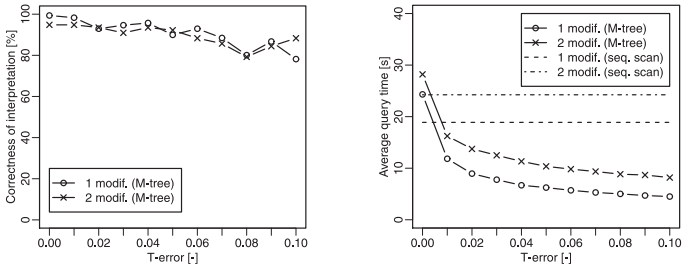


Figure 7.4: Speeding-up using M-tree (spectra with modifications) – a) correctness of interpretation and b) average query time

7.6 Advanced Analysis of Non-metric Similarity Search

The algorithm for identification of peptide sequences using the approximate similarity search by MAMs (Sec. 6.2) was additionally tested using the setup Keller 1 (Sec. 7.2.2) [154]. The d_{HP}, d'_{HP}, d_A and d'_A were compared, the M-tree was compared with LAESA and NM-tree, and the impact of different k in kNN queries on the efficiency and effectiveness of the search was also studied.

7.6.1 Comparison of d_{HP}, d'_{HP}, d_A and d'_A

The indexability of d_{HP} ($n = 50$) was analyzed when FP-bases and RBQ-bases proposed in Tab. 7.3 were used. In Fig. 7.5a,b observe the impact of T-error tolerance θ on the distance distributions obtained using the TriGen-modified d_{HP}, considering either FP-base or RBQ-bases. Obviously, a higher θ leads to a more convex T-modifier, and so to lower ρ (distance distribution pushed to the left). In Fig. 7.5c, see the same for the angle distance d_A. In fact, about 35% of all pairwise distances are in the maximum distance $d_A = 1$ (not shown for all T-error tolerances in Fig. 7.5c for better readability). These 35% distances cannot be fixed by the TriGen algorithm because they are indistinguishable.

In Fig. 7.5a,c, the distances d'_{HP} (Eq. 6.12) and d'_A (Eq. 6.11) are shown for a comparison with d_{HP} and d_A (the TriGen was not employed because T-error $\doteq 0$; $c_1 = 1$, $c_2 = 1$ for both d'_{HP} and d'_A). Since the indexability of d'_{HP} and d'_A is good, the average query time and the correctness of interpretation were measured on DB_2 indexed by M-tree where 1000NN queries and Q_2 were used. The average

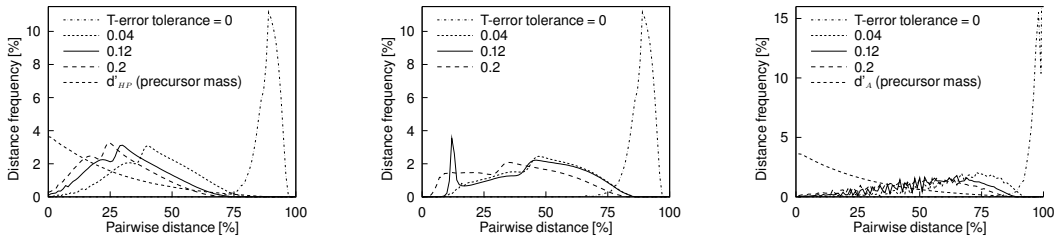


Figure 7.5: Distance distribution histograms – a) d_{HP} with FP-base and d'_{HP}, b) d_{HP} with RBQ-bases, c) d_A with FP-base and d'_A

query time was only 0.4 s for d'_A and d'_{HP}. The d'_{HP} with the M-tree was 33× faster than the sequential scan. It was also 14.9× faster than d_{HP} with M-tree when $\theta = 0.1$ (see the curve for 10^3NN in Fig. 7.9a for a comparison with d_{HP} for different θ).

The correctness of interpretation was 89.6% for d'_A and 85.7% for d'_{HP}. Since the search is approximate, the correctness was only 57.8% for d_{HP} when $\theta = 0.1$ (see the curve for 10^3NN in Fig. 7.8a for a comparison with d_{HP} for different θ). Although the indexability, the correctness of identification and the speed up of d'_{HP} were good on the M-tree, an extension of d'_{HP} (and d'_A) for the identification of peptides from query spectra containing modifications might be limited because d'_{HP} aggregates d_{HP} with the difference of precursor masses of compared spectra.

7.6.2 Comparison of M-tree with LAESA

The M-tree and LAESA (pivot table) utilizing the d_{HP} with FP-base were compared to the sequential scan of entire database. The experiments were performed on DB_1 and Q_1, 2000NN queries were used, and 8 processor cores were employed. The LAESA was constructed for 40 randomly selected pivots. The distance computations ratio got lower (w.r.t. seq. scan) when the T-error tolerance θ was higher. The lowest numbers of distance computations were obtained using LAESA for $\theta = 0.04$ and higher (Fig. 7.6c). However, the best average query time was obtained using M-tree (Fig. 7.6b). Since LAESA was stored in the main memory, it was also fast, but the size of the pivot table was almost 2 GB. The correctness of interpretation was lower with increasing T-error tolerance θ for both – M-tree and LAESA (Fig. 7.6a).

The performance of M-tree and LAESA (having 50 pivots) with d_{HP} was also tested on databases of different sizes generated from DB_1 (from 10 to 100 thousands of proteins; from 650 thousands to 5.6 millions of peptides or indexed vectors). The Fig. 7.7a shows the impact of the database size on the average query time, while $\theta = 0.1$. The LAESA is faster than the M-tree as long as all its blocks are stored in the main memory. If the main memory size is exceeded, the LAESA becomes inefficient. A 600 MB buffer was allocated in the main memory and it was exceeded when 25 thousands of protein sequences (1.5 millions of peptides) were indexed by LAESA. Moreover, for more than 40 thousands of protein sequences (2.4 millions of peptides) the sequential scan outperformed the LAESA. Fig. 7.7b shows that distance computations are misleading for LAESA when the size of the main memory is exceeded.

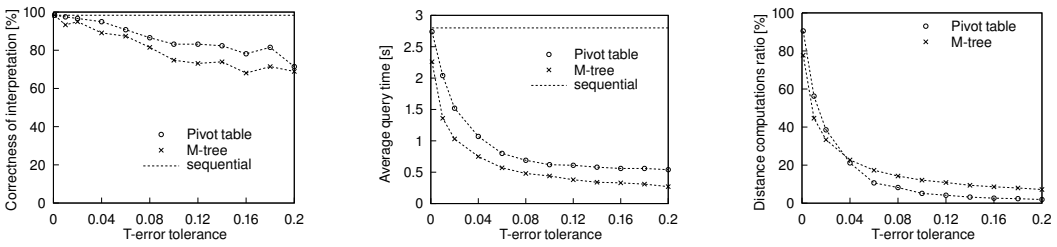


Figure 7.6: Comparison of M-tree and LAESA – a) correctness of interpretation, b) average query time, c) distance computations ratio

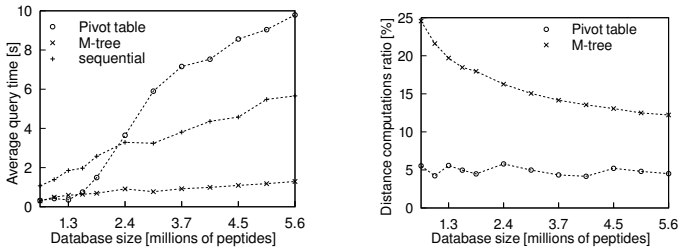


Figure 7.7: Database size – a) average query time, b) distance computations ratio

7.6.3 k in kNN queries

The following experiments were performed on M-tree while DB_2 and Q_2 were utilized. The correctness of mass spectra interpretation is lower with increasing T-error tolerance θ. However, the kNN queries with higher k can be used to avoid this problem (Fig. 7.8a). The correct peptide sequences are not spread uniformly over all interval $\langle 1..k \rangle$ of a kNN query result set but they are cumulated among a few nearest neighbors in many cases. As shown in Fig. 7.8b, the first nearest neighbor taken from the 100NN result was more likely to be correct than when taking the first nearest neighbor from 10NN result. The average query time of a kNN query and its distance computations ratio (w.r.t. sequential scan) increases with k (Fig. 7.9).

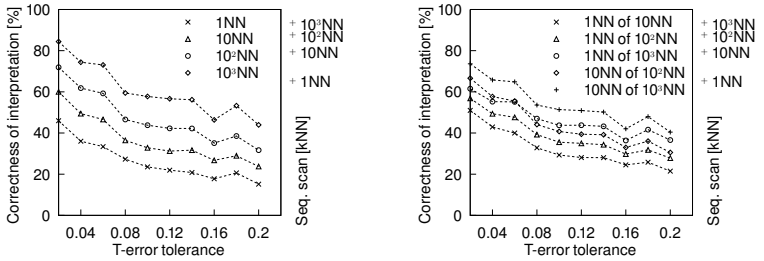


Figure 7.8: k in kNN queries – correctness of interpretation (d_{HP})

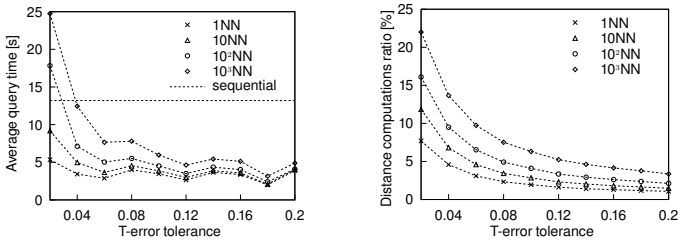


Figure 7.9: kNN queries – a) average query time, b) distance computations ratio

7.6.4 Comparison of a set of M-trees with NM-tree

Since a new M-tree structure must created for each T-error tolerance θ, the utilization of NM-tree was tested [163]. The NM-tree has been designed to be able

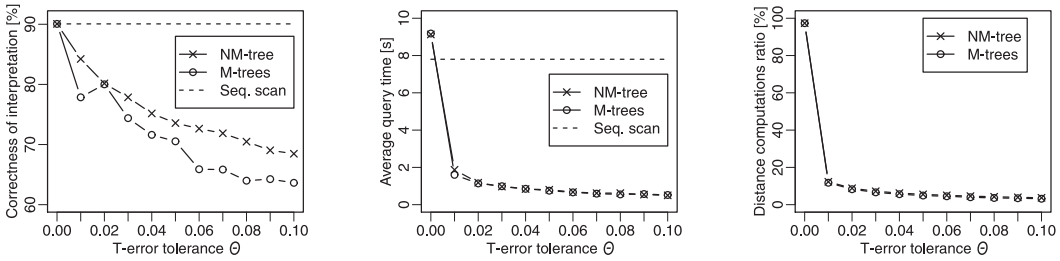


Figure 7.10: Comparison of the NM-tree with the set of M-trees – a) correctness of interpretation, b) average query time, c) distance computations ratio

to change θ at the query time. Another words, there is no necessity to re-index the NM-tree when θ is changed. The NM-tree was compared to the set of M-trees for different θ when 1000-NN queries were used. The database DB_1 and the query set Q_2 were used (Sec. 7.2.2). Note that the mass range of generated fragment ions was 300-2,000 Da what rapidly increases the speed up (decreases ρ).

The average query time and distance computations ratio (number of d_{HP} calls w.r.t. sequential scan) for the NM-tree were almost the same as for the set of M-trees (Fig. 7.10b,c). The NM-tree was 15.6× faster than the sequential scan ($\theta = 0.1$). Generally, the search is faster with increasing θ, while the correctness of interpretation w.r.t. the sequential scan is lower (Fig. 7.10a). The correctness is better for the NM-tree than for the set of M-trees with increasing θ.

The computation of modified distances in the NM-tree can be expensive (Sec. 5.2.5) and it can degrade the overall NM-tree's performance. Nevertheless, this can be solved by a table of precomputed modified distances. Moreover, computation of $d_{HP}^{f_M}$ and $f_M^{-1}(d_{HP}^{f_M})$ can be omitted for the purposes of mass spectra interpretation because the d_{HP} is already a metric distance (T-error $\doteq 0$).

7.7 Clustering and Sequential Scan of Protein Sequence Candidates

The optimization of peptide/protein sequences identification by clustering and a sequential scan of protein sequence candidates (Sec. 6.4) has been tested using the setup Keller 2 (Sec. 7.2.3) [155]. The optimization is suitable for small mixtures of purified proteins where a pre-processing by the clustering significantly speeds up the search and a post-processing by the sequential scan of protein sequence candidates significantly increases the number of identified peptide sequences.

The MAM used in the query phase is the non-metric tree (NM-tree) [152] because it combines the M-tree with the TriGen algorithm in a way that allows to dynamically control the retrieval precision at query time, i.e., the NM-tree does not have to be re-indexed for each θ. Note that the NM-tree can be replaced by any other MAM, because the approach is independent on a specific method.

The number of clusters corresponds to the number of those containing at least 2 spectra. Since one kNN query is performed per cluster, the number of clusters determines the number of kNN queries processed by the NM-tree. The number of missed spectra is counted after the clustering phase and before query phase

(Fig. 6.7). It corresponds to the number of annotated spectra in clusters with single objects and thus missed by clustering.

Single runs means that query sets of spectra from more spectrometer runs were processed separately and the results were summed (the number of clusters, the number of missed spectra, the time of clustering and the ratio of identified spectra to annotated spectra) or averaged (the time of identification per spectrum). *Appended runs* means that query sets of spectra from more spectrometer runs were processed together.

7.7.1 Clustering of Spectra from Two Spectrometer Runs

The clusters formed from appended query sets of spectra from two spectrometer runs contain many more annotated spectra than clusters formed from the query sets which are processed separately (Tab. 7.5). On average, the clusters formed from spectra from two spectrometer runs contain about 40.7% more annotated spectra than clusters formed from a single spectrometer run. Since one kNN query is performed per cluster containing at least 2 spectra, up to 79% of all kNN queries are not performed for the clusters formed from the spectra appended from two runs. For clusters formed from the spectra from single runs, up to 87% of all kNN queries are not performed but there are many missed annotated spectra.

Query set	Num. of all spectra	Num. of annotated spectra	Single runs			Appended runs		
			Num. of clusters	Spectra missed	Clustering time [s]	Num. of clusters	Spectra missed	Clustering time [s]
A1-2	2213	215	321	92	7.9	397	16	16.3
A3-4	1858	158	304	69	5.3	400	25	10.4
A5-6	2021	164	306	49	6.5	385	18	13.7
B1-2	618	121	49	87	0.5	113	33	1.0
B3-4	902	155	86	104	1.1	203	9	2.2
B5-6	1418	208	185	122	3.0	313	23	5.8
B7-8	1661	223	212	123	4.2	365	13	8.0

Table 7.5: Clustering of spectra from single runs and from two appended runs

7.7.2 Effectiveness and Efficiency of Identification

The impact of the clustering of query spectra on the number of finally identified peptide sequences (i.e., after the post-processing) and on the average time of identification per spectrum was tested. The sequential scan of the entire database and the NM-tree were compared in 3 different ways – without the clustering, with the clustering of two query sets processed separately and with the clustering of a query set appended from two query sets. When the clustering and/or the NM-tree were employed, the post-processing was used.

The most peptide sequences (on average 94.6%) were identified when the sequential scan was performed without the clustering (Tab. 7.6). On average 93.8% peptide sequences were identified when the NM-tree was employed without the clustering. The ratio of identified peptides was noticeably worse when the clustering was applied on the query sets from single runs – about 75.3% for the sequential scan and only 65.4% for the NM-tree. When the clustering was applied on the query sets appended from two spectrometer runs, the ratio of

Query set	Without clustering		With clustering			
			Single runs		Appended runs	
	Seq. scan	NM-tree	Seq. scan	NM-tree	Seq. scan	NM-tree
A1-2	96.7	96.7	74.0	72.6	95.8	95.8
A3-4	91.1	90.5	69.6	59.5	88.6	81.6
A5-6	93.3	92.7	81.1	78.7	93.3	87.2
B1-2	98.3	95.9	59.5	28.9	97.5	87.6
B3-4	97.4	96.8	81.3	71.0	97.4	96.8
B5-6	91.8	90.9	87.0	78.8	90.9	90.9
B7-8	93.3	93.3	74.4	68.2	91.9	91.0

Table 7.6: The ratio of identified spectra to annotated spectra [%]

Query set	Without clustering		With clustering			
			Single runs		Appended runs	
	Seq. scan	NM-tree	Seq. scan	NM-tree	Seq. scan	NM-tree
A1-2	7.36	0.33	1.13	0.05	1.42	0.08
A3-4	7.09	0.30	1.27	0.05	1.63	0.08
A5-6	7.28	0.30	1.17	0.05	1.53	0.07
B1-2	6.75	0.23	0.59	0.02	1.38	0.06
B3-4	6.92	0.24	0.73	0.03	1.75	0.07
B5-6	6.94	0.27	0.99	0.04	1.70	0.08
B7-8	7.20	0.30	0.97	0.04	1.73	0.08

Table 7.7: Average time of identification per spectrum [s]

identified peptides was almost the same like when no clustering was employed. On average, it was about 93.6% for the sequential scan and 90.1% for the NM-tree. The clustering of query sets appended from 2 runs worsened the ratio of identified peptides about 1% when the sequential scan was performed over the entire database and about 3.7% when the NM-tree was employed.

The slowest method was the sequential scan without the clustering where the average time of identification per spectrum was 7.04 s (Tab. 7.7). The NM-tree without the clustering took 0.28 s thus the speed-up was 25.1×. When the clustering was applied on the query sets from single runs, the average time was 0.98 s (speed-up 7.2×) for the sequential scan and 0.04 s (speed-up 176.0×) for the NM-tree. When query sets from two spectrometer runs were appended and the clustering was applied, the average time was 1.59 s for the sequential scan (speed-up 4.4×) and 0.07 s for the NM-tree (speed-up 100.6×). When the NM-tree was employed with the clustering, the average speed-up was 4× w.r.t. NM-tree without the clustering.

7.7.3 Clustering of Spectra Appended from More Runs

The impact of the increasing number of spectra from more spectrometer runs in a query set on the number of annotated mass spectra missed by clustering and on the time of clustering (Tab. 7.8) was analyzed. We can observe that the number of missed annotated spectra is almost the same when spectra from two or more spectrometer runs are appended, thus appending spectra from more than two spectrometer runs does not significantly improve the effectiveness of peptide sequences identification. Since we employ a simple clustering algorithm (Alg. 16), a disadvantage of appending spectra from too many spectrometer runs is that the time of clustering increases with the quadratic time complexity.

Query set	Num. of all spectra	Num. of annotated spectra	Num. of clusters	Ratio of clust. to all spectra [%]	Spectra missed	Clustering time [s]	Ratio of identified spectra [%]	Avg. time of ident. [s]
A1	1122	119	157	14.0	49	4.0	76.5	0.05
A1-2	2213	215	397	17.9	16	16.3	95.8	0.08
A1-3	3038	274	540	17.8	15	30.6	96.0	0.08
A1-4	4071	373	706	17.3	15	61.2	94.4	0.09
A1-5	5071	451	839	16.5	16	106.4	94.9	0.09
A1-6	6092	537	943	15.5	17	148.3	94.0	0.09

Table 7.8: Clustering of spectra appended from more spectrometer runs

The ratio of identified to annotated spectra and the average time of identification per spectrum were also measured on the NM-tree. The ratio of identified spectra is almost the same when spectra from two or more spectrometer runs are appended (on average 95%). The time of identification a bit increases with increasing number of spectra because of the quadratic time complexity of clustering.

We can observe that the ratio of the number of clusters to the number of all spectra in a query set is lower with the increasing number of spectra. This could be an advantage for large query sets of mass spectra because only a small number of the spectra is queried and thus the search is significantly faster. When spectra from 14 spectrometer runs on mixture A were appended, 14365 spectra formed 1188 clusters with more than one spectrum. Thus only 8.3% of all queries were performed on the NM-tree. When spectra from 8 spectrometer runs on mixture B were appended, 4599 spectra formed 711 clusters thus only 15.5% of all queries were performed.

7.7.4 Impact of Distance Threshold on Clustering

The impact of the threshold t of d_{HP} on the number of clusters, on the number of spectra missed by the clustering and on the time of clustering was tested (Tab. 7.9). The dataset A1-2 with 2213 spectra appended from two spectrometer runs from the setup Keller 2 (Sec. 7.2.3) was used. The number of clusters is bigger with increasing t while the number of spectra missed by clustering is smaller. The optimal t seems to be about 0.65 when the number of clusters (or kNN queries performed, respectively) is only 17.9% w.r.t. the number of kNN queries which must be performed when the clustering is not employed. Moreover, there are only 16 missed spectra. For $t < 0.65$, the number of spectra missed by clustering grows because there are less hits among the theoretical and the query spectra. The ratio of identified to annotated spectra is still more than 95% because the sequential scan of protein sequence candidates is employed. For $t > 0.65$, the number of clusters increases (up to $t = 0.75$) and the number of missed spectra is almost zero. A disadvantage is that high t may form clusters of spectra not coming from the same peptide. In practice, the optimal t depends on the number of peaks in query spectra. The optimal t may be higher than 0.65 when the support of modifications is implemented as described in Sec. 6.3. The time of identification a bit increases with the increasing t – this corresponds to the increasing number of clusters.

t	Num. of clusters	Spectra missed	Clustering time [s]	Ratio of ident. spectra [%]	Avg. time of ident. [s]
0.3	162	93	16.6	93.0	0.04
0.4	242	69	16.3	95.3	0.05
0.5	312	49	16.9	95.3	0.06
0.6	368	31	16.0	95.8	0.07
0.65	397	16	16.3	95.8	0.08
0.7	562	4	17.0	95.8	0.11
0.75	633	0	18.0	95.8	0.12
0.8	318	0	24.8	49.8	0.07

Table 7.9: Impact of distance threshold t on clustering

7.8 Utilization of Precursor Mass Filter

Even though the identification of peptide sequences by non-metric access methods has been successfully tested on query sets containing small mixtures of purified proteins (Sec. 7.7), their utilization on query sets of complex protein mixtures containing thousands of proteins is a non-trivial task. Thus the precursor mass filter (Sec. 4.1) has been also implemented as a database indexing technique to support complex query sets [160]. When the precursor mass filter is used, d_{HP}^{match} outperforms several state-of-art tools for peptide sequences identification from HPLC-MS/MS spectra in both – the number of identified peptides and in the speed of search.

The identification of peptide sequences using the precursor mass filter followed by a ranking of theoretical spectra by d_A, d_{HP} and d_{HP}^{match} (Eq. 6.10) was tested in the following experiments (i.e., MAMs were not employed). The setups E. coli and Human were used for this purpose (Sec. 7.2.4). The results were compared with state-of-the-art tools OMSSA [61] and X!Tandem [62]. Peptide identifications from all engines were statistically evaluated by OpenMS v. 1.9 [93]. When a fixed modification was searched, the mass of an amino acid was changed when theoretical spectra were generated, e.g., the mass of *cysteine* was increased by approx. 57.02 Da. When a variable modification was searched, theoretical spectra with all possible shifts of $\frac{m}{z}$ values were generated, compared with the query spectrum and the theoretical spectrum with the best score was selected to form a PSM.

7.8.1 State-of-the-Art Tools

First of all, a comparison of the state-of-the-art tools was performed. The numbers of identified peptides for different q-values (Sec. 3.2.3) and search times were measured using the freely available tools OMSSA (v. 2.1.8) and X!Tandem (v. 2011.12.01.1). The refinement mode in X!Tandem was not used because it impacted the statistical evaluation. The comparison was made using OpenMS (v. 1.9). Simple pipelines in TOPPAS were created for this purpose (e.g., *OMSSAAdapter* → *PeptideIndexer* → *FalseDiscoveryRate* → *IDFilter*, where *OMSSAAdapter* calls OMSSA search engine, *PeptideIndexer* annotates for each search result whether it is a target or a decoy hit, *FalseDiscoveryRate* computes q-values and *IDFilter* selects only those PSMs with q-values less or equal a specified tolerance). Pipelines were processed without and with the support of modifications (*carbamidomethylation of cysteine* was used as a fixed modification and

		OMSSA				X!Tandem			
		q-value			Time	q-value			Time
		0.05	0.01	0.001		0.05	0.01	0.001	
E.coli		11,729	10,301	7,989	03:40	10,277	8,518	6,398	03:35
	mod.	13,008	11,435	9,123	05:00	11,612	9,813	7,487	04:36
Hum48		6,893	6,147	**5,439**	27:42	7,330	5,902	4,239	40:03
	mod.	9,430	8,508	**7,229**	27:42	**10,494**	8,524	5,411	51:20
Hum49		8,118	7,119	**5,392**	24:03	7,728	6,085	4,673	37:07
	mod.	11,465	10,333	8,601	25:10	11,695	9,712	6,119	53:30

		d_A				d_{HP}			
		q-value			Time	q-value			Time
		0.05	0.01	0.001		0.05	0.01	0.001	
E.coli		12,846	10,404	1,785	00:40	13,004	11,200	8,307	00:44
	mod.	14,554	11,587	1,948	01:26	14,576	12,556	9,288	01:33
Hum48		6,205	3,717	773	03:19	7,162	5,863	4,225	03:53
	mod.	8,615	4,845	934	06:06	9,882	8,108	6,119	06:29
Hum49		7,859	5,186	2,512	03:05	8,347	6,888	5,329	03:44
	mod.	11,244	6,944	3,291	05:12	12,002	10,011	8,120	06:46

		d_{HP}^{match}			
		q-value			Time
		0.05	0.01	0.001	
E.coli		**13,373**	**11,668**	**9,402**	00:43
	mod.	**14,969**	**13,113**	**10,005**	01:30
Hum48		**7,461**	**6,230**	4,497	03:42
	mod.	10,305	**8,680**	7,109	06:34
Hum49		**8,760**	**7,512**	5,311	03:57
	mod.	**12,531**	**10,816**	**8,689**	06:15

Table 7.10: Numbers of identified peptides and search times [min:sec]. In a cell with a number of identified peptides, the best result among all engines is highlighted.

oxidation of methionine as a variable modification). The results are shown in Tab. 7.10. OMSSA identified more peptides than X!Tandem in all query sets and the search was 1.5× - 2.1× faster on human query sets.

7.8.2 SimTandem

Numbers of peptides identified by SimTandem (i.e., by precursor filter followed by a ranking of theoretical spectra by d_A, d_{HP} and d_{HP}^{match}) and search times are shown in Tab. 7.10. The most peptides are identified when d_{HP}^{match} is used. However, when $q = 0.001$, OMSSA identifies more peptides in three cases. When $q = 0.05$, X!Tandem identifies the most peptides in one case. d_{HP}^{match} identifies more peptide sequences than d_{HP} in almost all cases. The number of identified peptides is bigger for d_{HP} than for X!Tandem when *E.coli* and *Hum49* are used. OMSSA outperforms d_{HP} on human query sets when $q \leq 0.01$ is used. The number of identified peptides is significantly smaller for d_A than for other engines.

SimTandem is 5.0×-7.1× faster than OMSSA without the support of modifications and 3.2×-4.3× faster with the support of modifications. It is also 4.9×-10.3× faster than X!Tandem without modifications and 3.0×-7.9× faster with modifications. The speed up is almost the same when d_{HP}^{match} is used.

A graphical comparison of numbers of identified peptides with state-of-the-art tools is shown in Fig. 7.11. The overlap of identified peptides between SimTandem and OMSSA is bigger than the overlap between SimTandem and X!Tandem and also bigger than the overlap between OMSSA and X!Tandem on all query sets.

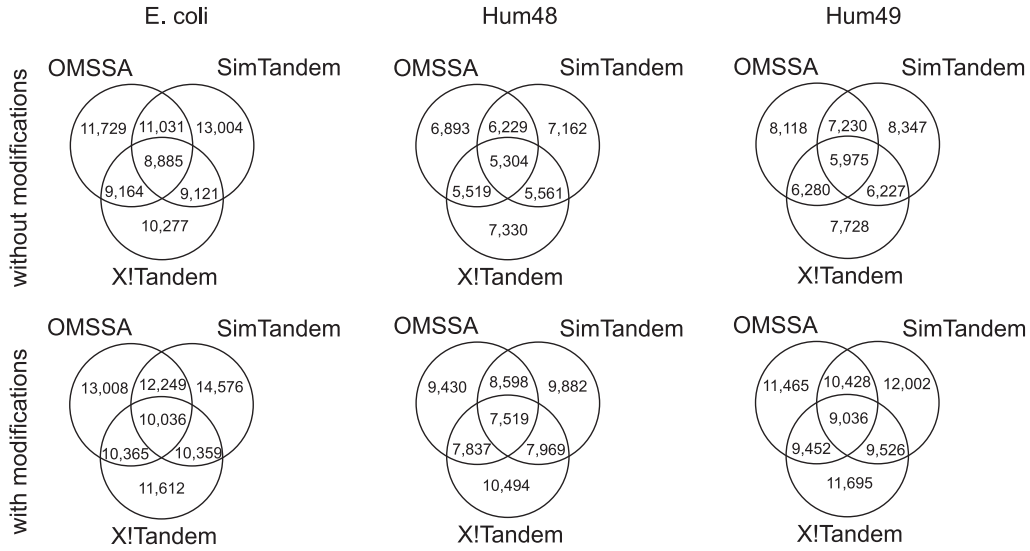


Figure 7.11: Comparison of d_{HP} with state-of-the-art tools ($q = 0.05$)

We have tested the impact of the index of the root n in d_{HP} and d_{HP}^{match} on the number of identified peptides. Results are shown in Tab. 7.11. The number is bigger with bigger n. However, when n is too big, the number of identified peptides is smaller. The optimal n depends on the data sets and should be determined empirically.

7.8.3 Efficiency of Precursor Mass Filter

The efficiency of the precursor mass filter was also studied when different precursor mass error tolerances λ and different protein sequence databases were utilized. Since the number of comparisons of a query spectrum with theoretical spectra is crucial for the efficiency of the precursor mass filter, an average number of comparisons was measured in protein sequence databases Swiss-Prot (v. 07/2012) (human sequences only and all sequences) [3], MSDB [2] and NCBI RefSeq (v. 55) [4]. The query set *Hum48* was used and modifications were not supported. The results are shown in Tab. 7.12. Since an organism is usually known for a query set of spectra (e.g., E. coli or human) and the precision of modern spectrometers increases, the number of spectra compared with a query

n	d_{HP}			d_{HP}^{match}		
	E.coli	*Hum48*	*Hum49*	*E.coli*	*Hum48*	*Hum49*
5	4,237	1,605	2,321	8,307	3,283	4,646
10	6,288	2,590	4,167	9,165	4,194	**5,517**
20	8,101	3,773	5,160	9,360	4,440	5,220
30	8,307	4,225	5,329	9,402	4,497	5,311
50	8,370	4,343	5,144	9,427	4,530	5,268
100	**8,415**	**4,447**	**5,579**	**9,453**	**4,560**	5,295
∞	8,348	3,928	5,023	9,332	3,988	5,110

Table 7.11: Numbers of identified peptides by d_{HP} and d_{HP}^{match} for different n ($q = 0.001$, modifications in query spectra were not supported). The best result in each column is highlighted.

spectrum is small and thus the precursor filter is efficient. For example, 409 spectra are compared with a query spectrum when human sequences from Swiss-Prot are used and when $\lambda = 10\,\text{ppm}$. When the NCBI database is used, the number of comparisons is 60,638. For a mass spectrometer with a low precision $\lambda = 2\,\text{Da}$, the number of comparisons is significantly bigger. For example, 15,564 spectra are compared when human sequences from Swiss-Prot are used and 2,451,235 comparisons are made when the NCBI database is used.

Database	Number of proteins	Number of peptides	λ					
			5 ppm	10 ppm	15 ppm	0.5 Da	1 Da	2 Da
Swiss-Prot (human)	173,450	9,567,012	206	409	613	3,892	7,792	15,564
Swiss-Prot (complete)	1,073,578	52,361,610	1,056	2,091	3,135	21,261	42,645	85,141
MSDB	6,478,158	281,767,270	5,756	11,369	17,042	113,272	227,017	453,153
NCBI	34,737,538	1,533,987,691	30,606	60,638	91,004	612,225	1,227,339	2,451,235

Table 7.12: Average numbers of comparisons of a query spectrum with theoretical spectra for different protein sequence databases and different precursor mass error tolerances λ. Numbers of protein and peptide sequences in databases are also proposed (decoy sequences are included in the databases).

Chapter 8

Implementation

SimTandem is a freely available tool for identification of peptides from HPLC-MS/MS spectra. It is based on the similarity search of query mass spectra in a database of theoretical spectra generated from a database of known protein sequences. SimTandem employs the parameterized Hausdorff distance as the mass spectra similarity function. SimTandem has been implemented in two alternatives. First, it has been implemented as the on-line web application to demonstrate the utilization of non-metric access methods for the fast and approximate similarity search in a database of theoretical tandem mass spectra. Second, SimTandem has been implemented as a peptide identification tool which can be used in the framework TOPP [104] based on OpenMS [93]. The advantages of the TOPP tool are that the results can be statistically evaluated, easily compared with other state-of-the-art tools, and that many other tools for processing of mass spectra can be used as SimTandem's predecessors or successors. The TOPP tool supports the precursor mass filter next to the approach based on (non)metric access methods. The SimTandem is freely available at [166].

8.1 Web Interface

The on-line version of SimTandem demonstrates the utilization of MAMs as database indexing techniques for identification of peptide/protein sequences from HPLC-MS/MS spectra [166]. The index is created over theoretical spectra generated from a database of protein sequences. The core of the application is implemented in C++ and it employs the Siret Object Library (SOL) – a framework for

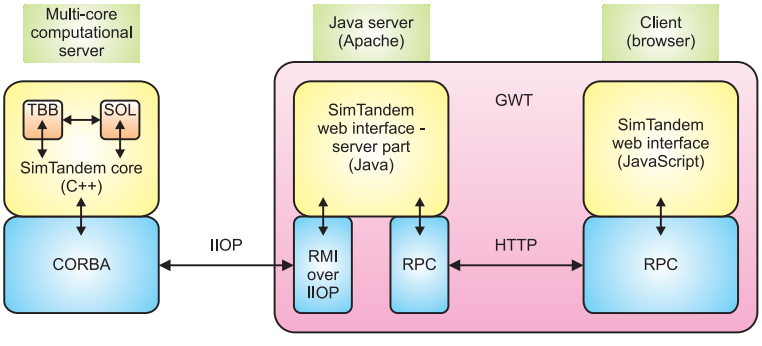


Figure 8.1: SimTandem (on-line version) – architecture

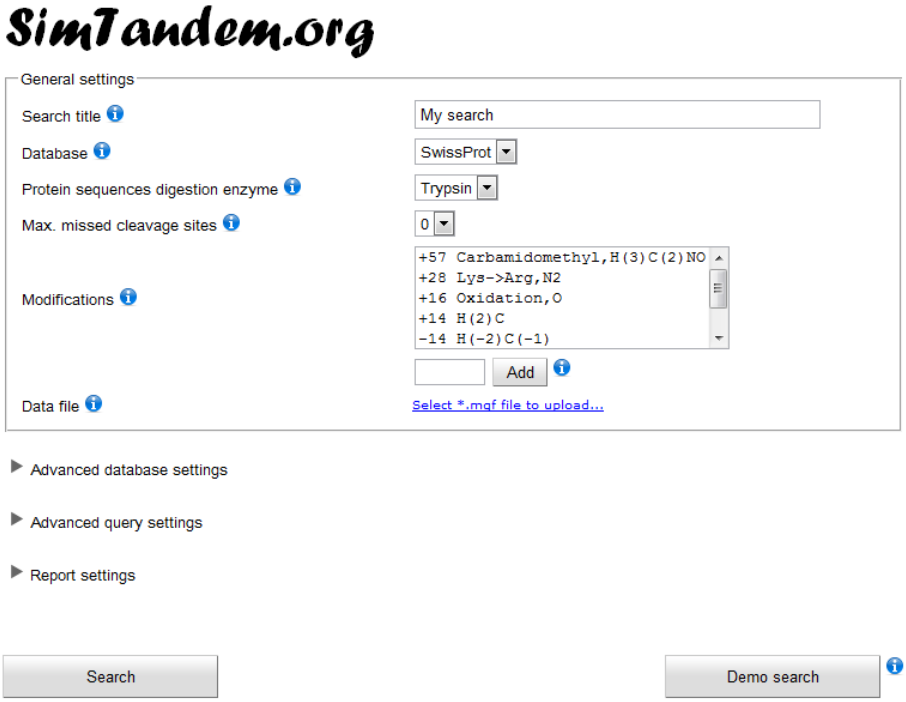


Figure 8.2: SimTandem (on-line version) – general settings

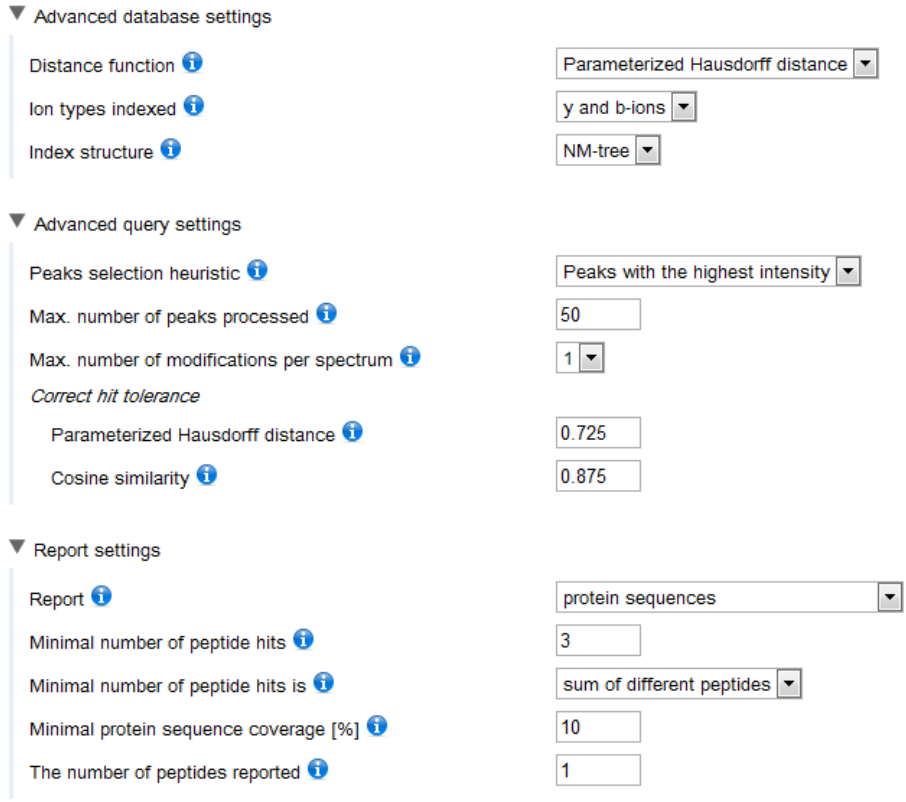


Figure 8.3: SimTandem (on-line version) – advanced options

Protein Identification Summary:

Proteins searched: 526969
Peptides searched: 30685369
Proteins found: 21
Spectra tested: 1122

1. protein: sp|P00921|CAH2_BOVIN
Protein size: 260
Digested peptides: 36
Sequence coverage: 71.154%

MSHHWGYGKH NGPEHWHKDF PIANGERQSP VDIDTKAVVQ DPALKPLALV YGEATSRRMV NNGHSFNVEY DDSQDKAVLK DGPLTGTYRL VQFHFHWGSS
DDQGSEHTVD RKKYAAELHL VHWNTKYGDF GTAAQQPDGL AVVGVFLKVG DANPALQKVL DALDSIKTKG KSTDFPNFDP GSLLPNVLDY WTYPGSLTTP
PLLESVTWIV LKEPISVSSQ QMLKFRTLNF NAEGEPELLM LANWRPAQPL KNRQVRGFPK

Matched peptides: 15

YGDFGTAAQQPDGLAVVGVFLK
Start: 126; End: 147
Matched spectra: 4

YAAELHLVHWNTKYGDFGTAAQQPDGLAVVGVFLK
Start: 113; End: 147
Matched spectra: 3

MVNNGHSFNVEYDDSQDK
Start: 58; End: 75
Matched spectra: 2

VGDANPALQK
Start: 148; End: 157
Matched spectra: 2

Spectrum ID: 22
d_{HP}: 0.503
d_A: 0.688
Matched ions: 23

		a	b	b-H_2O	b-NH_3	y	y-H_2O	y-NH_3 / z	y-H_2O-H_2O	y-NH_3-NH_3	y-H_2O-NH_3	y^{2+}	y^{2+}-H_2O	y^{2+}-NH_3	y^{3+}		
1	V	72.081	100.076	82.066	83.050	-	-	-	-	-	-	-	-	-	-	V	10
2	G	129.103	157.098	139.087	140.071	913.474	895.464	896.448	877.453	879.421	878.437	457.241	448.236	448.728	305.163	G	9
3	D	244.130	272.125	254.114	255.098	856.453	838.442	839.426	820.432	822.400	821.416	428.730	419.725	420.217	286.156	D	8
4	A	315.167	343.162	325.151	326.135	741.426	723.415	724.399	705.405	707.373	706.389	371.217	362.212	362.704	247.814	A	7
5	N	429.210	457.205	439.194	440.178	670.389	652.378	653.362	634.368	636.336	635.352	335.698	326.693	327.185	224.135	N	6
6	P	526.263	554.257	536.247	537.231	556.346	538.335	539.319	520.325	522.293	521.309	278.677	269.672	270.164	186.121	P	5
7	A	597.300	625.295	607.284	608.268	459.293	441.283	442.267	423.272	425.240	424.256	230.150	221.145	221.637	153.770	A	4
8	L	710.384	738.379	720.368	721.352	388.256	370.245	371.229	352.235	354.203	353.219	194.632	185.627	186.119	130.091	L	3
9	Q	838.442	866.437	848.427	849.411	275.172	257.161	258.145	239.151	241.119	240.135	138.090	129.085	129.577	92.396	Q	2
10	K	-	-	-	-	147.113	129.103	130.087	111.092	113.060	112.076	74.061	65.055	65.547	49.710	K	1

Figure 8.4: SimTandem (on-line version) – identification of protein sequences

efficient metric and non-metric similarity search, which is currently being developed by SIRET Research Group (SRG) [168]. SimTandem uses Intel's Threading Building Blocks (TBB) [169] to support the parallel querying of large mass spectra query sets. The web interface is implemented in Java based Google Web Toolkit (GWT) [170]. The communication between the web interface and the core is realized using Java RMI over IIOP and CORBA [171]. A scheme of the architecture is shown in Fig. 8.1.

SimTandem's general settings (Fig. 8.2) are similar to the existing tools for mass spectra interpretation. A database of protein sequences, a protein sequences digestion enzyme and a maximum number of missed cleavage sites are set up. Since SimTandem supports the search of spectra with modifications, the user can define, which modifications will be supported during the search. Some of them are pre-defined but the user is allowed to add own modifications. A set of mass spectra can be uploaded in Mascot Generic Format (*.mgf).

Fig. 8.3 represents SimTandem's advanced options. Currently, the user can choose from two distances which the application supports – the d_{HP} and d_A. To speed-up the search, one can limit the maximum number of peaks in the queried spectra and the maximum number of modifications occurring simultaneously in a spectrum. The user can set the correct hit tolerance what is a maximum distance between a theoretical peptide mass spectrum generated from the database of

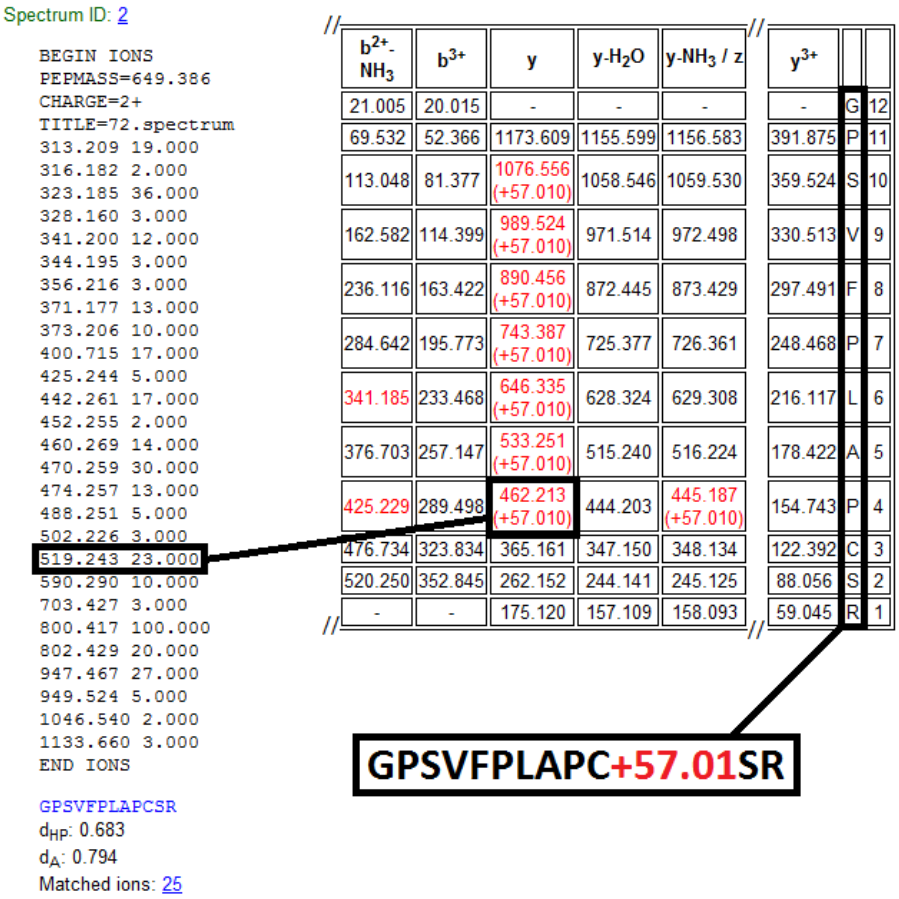


Figure 8.5: SimTandem (on-line version) – search of with modifications

protein sequences and a spectrum captured by a spectrometer. If a distance between the mass spectra is less or equal the correct hit tolerance, a hit between the peptide sequence and the mass spectrum is assumed.

The user can also customize the output report (Fig. 8.4). One can choose whether the query mass spectra will be reported independently or whether the spectra (peptide sequences, respectively) will be grouped into protein sequences. The user can set the minimum number of peptide sequence hits in a protein sequence and the minimum protein sequence coverage. Only the protein sequences achieving the both values are reported. The minimum number of peptide sequence hits can be determined as a sum of different peptide sequence hits in a protein sequence or as a sum of all peptide sequence hits (more spectra can match the same peptide sequence). The protein sequence coverage is the percentage of amino acids in the protein sequence, which are covered by peptide sequence hits. Finally, one can set the number of peptide sequences (k nearest neighbors), which will be reported for a spectrum.

In an output report, a table of fragment ions predicted from a peptide sequence is shown for a hit between a theoretical spectrum and a query spectrum. Fragment ions matched in the query spectrum (within a specified $\frac{m}{z}$ error tolerance ξ) and ions shifted by modifications are highlighted (Fig. 8.5).

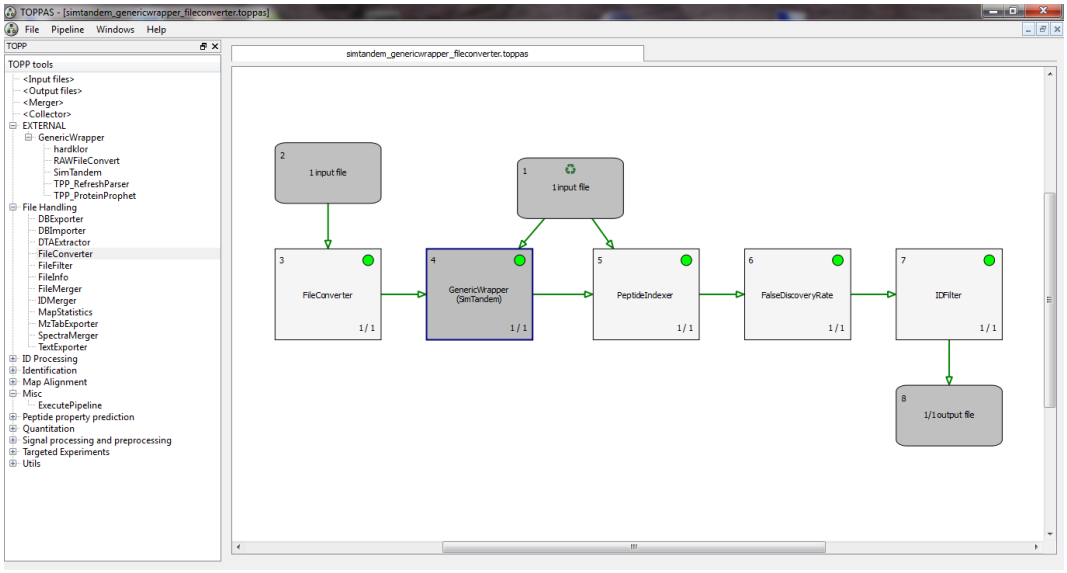


Figure 8.6: A simple identification pipeline in TOPPAS with SimTandem

8.2 TOPP Tool

SimTandem has been also implemented as a peptide identification tool which can be used by the framework TOPP [104] based on OpenMS [93]. Currently, the tool supports the precursor mass filter as a database indexing technique next to MAMs, thus complex mixtures containing thousands of proteins can be easily processed. The advantages of the TOPP tool are that:

- TOPP contains tools for statistical evaluation of PSMs and thus results can be easily evaluated and compared with other peptide identification tools like OMSSA [61], X!Tandem [62], MASCOT [60], MyriMatch [45], etc.;
- since pipelines formed from different tools can be created in TOPP, other tools in TOPP (or any executable file) can be used as SimTandem's predecessors or successors;
- since SimTandem is a stand-alone application, query sets from shotgun proteomics having sizes of hundreds of megabytes can be easily processed;
- SimTandem can be easily offered to end users via OpenMS.

A basic pipeline in TOPP, where SimTandem is utilized as a peptide sequences identification engine, is shown in Fig. 8.6. SimTandem requires an input *.mgf file with the query set of mass spectra and a *.fasta file with the database of protein sequences. The output is generated in the standardized *.IdXML file format. The configuration properties of SimTandem are shown in Fig. 8.7.

8.2.1 Installation Instructions

Below, the system requirements of the tool, the installation instructions for Windows x64 and an example of use are proposed.

System requirements Any 64-bit Windows system is suitable (Windows 7, Windows Server 2008, etc.). Other requirements are just recommended and depends on the sizes of the protein sequence databases and the sizes of query sets. Say, at least 4 to 8 GB RAM and a few GB of free disk space for temporary index files.

Installation

1. Download and install "OpenMS-1.9.0-Win_64bit_setup.exe" from the OpenMS website [93].
2. Download and unzip the lastest release of SimTandem from [166].
3. Put the file "SimTandem.ttd" into "../OpenMS-1.9/share/OpenMS/TOOLS/EXTERNAL".
4. Restart your computer.

Usage

1. Run TOPPAS.
2. SimTandem is available in the left menu "TOPP tools" in "EXTERNAL/GenericWrapper/SimTandem".
3. Drag & Drop SimTandem to the workspace.
4. Click with the right mouse button on the SimTandem's box and select "edit parameters".
5. Set the correct location of "simtandem_openms.exe" in the option "executable".
6. Optionally, the path in "indexDir" can be changed where the indexed files are stored (make sure that the path is writable).
7. Connect input files *.mgf and *.fasta with SimTandem and connect also the output *.IdXML file with an output box or with another tool.
8. If you have another input file than *.mgf, the tool "File Handling/FileConverter" can be used as SimTandem's predecessor.

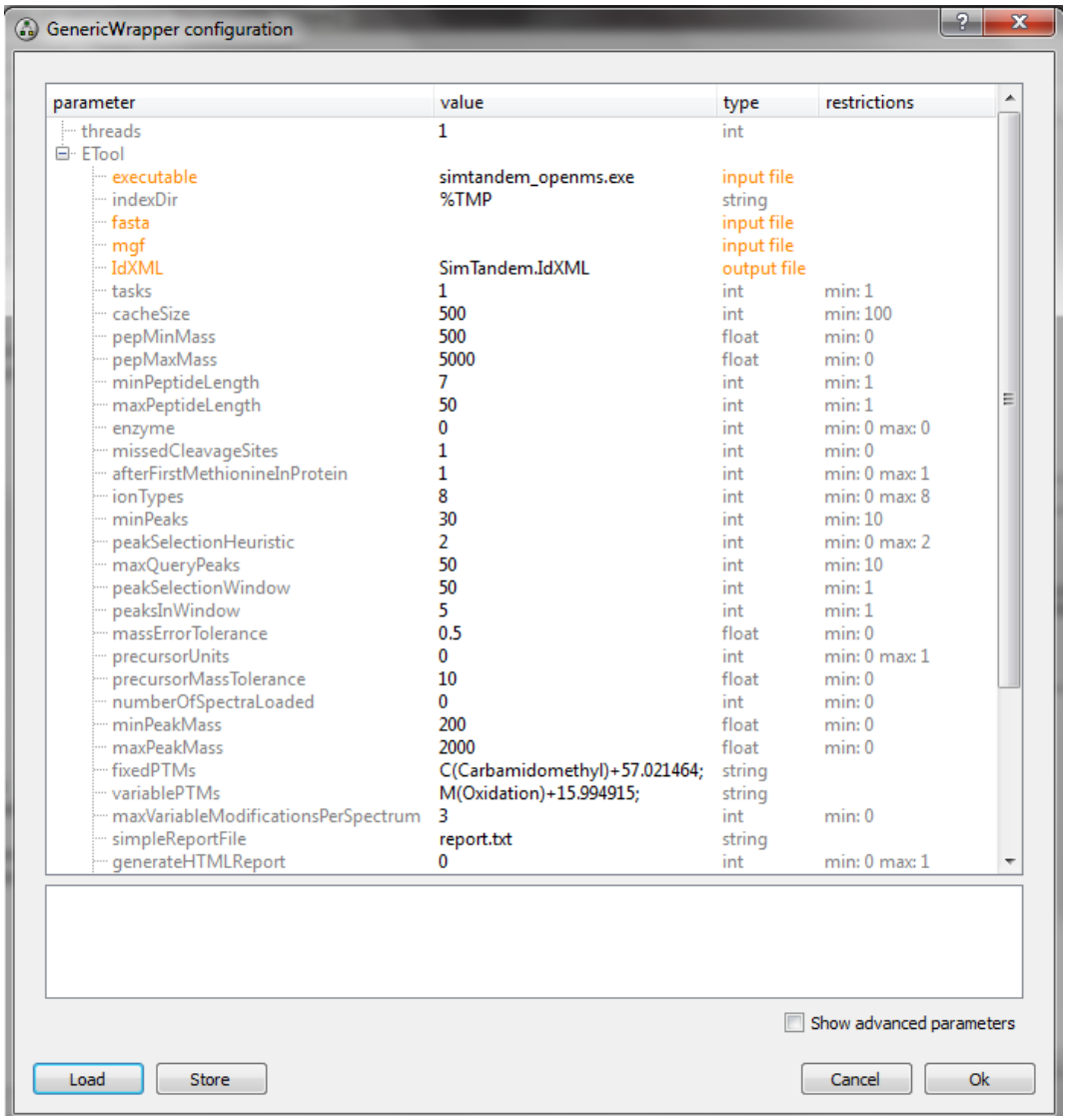


Figure 8.7: Configuration of SimTandem

Chapter 9

Conclusion

In this thesis, we studied the methods for identification of peptide sequences from tandem mass spectra based on the similarity search in databases of theoretical mass spectra generated from databases of known protein sequences. Since the sizes of protein sequence databases grow rapidly in recent years, we were focused on the database indexing techniques which can be employed to speed-up the search in databases of theoretical mass spectra.

We have shown that non-metric access methods can be used as the database indexing techniques for fast and approximate similarity search in databases of theoretical mass spectra. Currently, the proposed method is suitable for small mixtures of purified proteins (several tens of proteins). The method for peptide sequences identification is more than 100× faster than a sequential scan over the entire database of theoretical mass spectra while more than 90% of spectra (w.r.t. a ground truth determined by a state-of-the-art search engine) are correctly assigned to peptide sequences.

However, the utilization of non-metric access methods on complex mixtures of proteins obtained by a cell lysis (i.e., containing thousands of proteins) is a non-trivial task. To support the complex mixtures of proteins, we have also implemented the precursor mass filter as the database indexing technique. When the precursor mass filter is followed by a ranking of theoretical mass spectra using a modification of the parameterized Hausdorff distance (originally designed for non-metric indexes), our method outperforms state-of-the-art tools in both – the number of identified peptide sequences and the speed of search. The proposed methods have been implemented in the application SimTandem which can be used for a batch analysis in the framework TOPP based on OpenMS.

Bibliography

[1] Gregory Petsko and Dagmar Ringe. *Protein Structure and Function (Primers in Biology)*. New Science Press Ltd, London, UK, 2004.

[2] MSDB.
`http://www.proteomics.leeds.ac.uk/bioinf/`.

[3] UniProtKB/Swiss-Prot.
`http://www.uniprot.org/`.

[4] NCBI RefSeq.
`http://www.ncbi.nlm.nih.gov/RefSeq/`.

[5] Ingvar Eidhammer, Kristian Flikka, Lennart Martens, and Svein-Ole Mikalsen. *Computational Methods for Mass Spectrometry Proteomics*. John Wiley & Sons, England, 2007.

[6] Donald F. Hunt and et al. Protein Sequencing by Tandem Mass Spectrometry. In *Proc. Nati. Acad. Sci. USA*, volume 83, pages 6233–6237, 1986.

[7] Rune Matthiesen. *Mass Spectrometry Data Analysis in Proteomics (Methods in Molecular Biology)*. Humana Press, Totowa, New Jersey, 2007.

[8] Edmond de Hoffmann and Vincent Stroobant. *Mass Spectrometry: Principles and Applications*. John Wiley & Sons, England, 2007.

[9] Jesper V. Olsen, Shao-En Ong, and Matthias Mann. Trypsin Cleaves Exclusively C-terminal to Arginine and Lysine Residues. *Molecular and Cellular Proteomics*, 3:608–614, 2004.

[10] Linda Switzar, Martin Giera, and Wilfried M. A. Niessen. Protein Digestion: An Overview of the Available Techniques and Recent Developments. *Journal of Proteome Research*, 12(3):1067–1077, 2013.

[11] Peptide Mass Spectrum Interpretation.
`http://www.weddslist.com/ms/`.

[12] Nico Pfeifer, Andreas Leinenbach, Christian G. Huber, and Oliver Kohlbacher. Improving Peptide Identification in Proteome Analysis by a Two-Dimensional Retention Time Filtering Approach. *Journal of Proteome Research*, 8(8):4109–4115, 2009.

[13] Beata Reiz, Attila Kertesz-Farkas, Sandor Pongor, and Michael P. Myers. Data Preprocessing and Filtering in Mass Spectrometry Based Proteomics. *Current Bioinformatics*, 7(2):212–220, 2012.

[14] John R. Yates III. Mass Spectral Analysis in Proteomics. *Annual Review of Biophysics and Biomolecular Structure*, 33:297–316, 2004.

[15] Quadrupole on Wikipedia.
`http://en.wikipedia.org/wiki/Quadrupole_mass_analyzer`.

[16] Orbitrap on Wikipedia.
`http://en.wikipedia.org/wiki/Orbitrap`.

[17] Thermo Scientific.
`http://www.thermoscientific.com`.

[18] William J. Henzela, Colin Watanabea, and John T. Stults. Protein Identification: the Origins of Peptide Mass Fingerprinting. *Journal of the American Society for Mass Spectrometry*, 14(9):931–942, 2003.

[19] Pavel A. Pevzner and et al. Efficiency of Database Search for Identification of Mutated and Modified Proteins via Mass Spectrometry. *Genome Research*, 11(2):290–299, 2001.

[20] UNIMOD.
`http://www.unimod.org/`.

[21] Chris Bauer, Rainer Cramer, and Johannes Schuchhardt. Evaluation of Peak-Picking Algorithms for Protein Mass Spectrometry. In *Data Mining in Proteomics*, volume 696 of *Methods in Molecular Biology*, pages 341–352. 2011.

[22] Eva Lange, Clemens Gropl, Knut Reinert, Oliver Kohlbacher, and Andreas Hildebrandt. High-Accuracy Peak Picking of Proteomics Data Using Wavelet Techniques. In *Pac Symp Biocomput.*, pages 243–254, 2006.

[23] Jussi Salmi, Tuula A. Nyman, Olli S. Nevalainen, and Tero Aittokallio. Filtering Strategies for Improving Protein Identification in High-throughput MS/MS Studies. *Proteomics*, 9:848–860, 2009.

[24] Bernhard Y. Renard and et al. When Less Can Yield More - Computational Preprocessing of MS/MS Spectra for Peptide Identification. *Proteomics*, 9:4978–4984, 2009.

[25] Wenjun Lin and et al. An Adaptive Approach to Denoising Tandem Mass Spectra. In *IEEE Bioinformatics and Biomedicine Workshops*, pages 89–94, 2010.

[26] Wenjun Lin, Fang-Xiang Wu, Jinhong Shi, Jiarui Ding, and Wenjun Zhang. An adaptive approach to denoising tandem mass spectra. *Proteomics*, 11(19):3773–3778, 2011.

[27] Kristian Flikka and et al. Improving the Reliability and Throughput of Mass Spectrometry-based Proteomics by Spectrum Quality Filtering. *Proteomics*, 6:2086–2094, 2006.

[28] Alexey I. Nesvizhskii and et al. Dynamic Spectrum Quality Assessment and Iterative Computational Analysis of Shotgun Proteomic Data. *Molecular and Cellular Proteomics*, 5:652–670, 2006.

[29] Kristian Flikka and et al. Implementation and Application of a Versatile Clustering Tool for Tandem Mass Spectrometry Data. *Proteomics*, 7:3245–3258, 2007.

[30] Ari M. Frank and et al. Clustering Millions of Tandem Mass Spectra. *Journal of Proteome Research*, 7(1):113–122, 2008.

[31] Jayson A. Falkner, Jarret W. Falkner, Anastasia K. Yocum, and Philip C. Andrews. A Spectral Clustering Approach to MS/MS Identification of Post-translational Modifications. *Journal of Proteome research*, 7(11):4614–4622, 2008.

[32] Ilan Beer, Eilon Barnea, Tamar Ziv, and Arie Admon. Improving Large-scale Proteomics by Clustering of Mass Spectrometry Data. *Proteomics*, 4:950–960, 2004.

[33] David L. Tabb and et al. Similarity among Tandem Mass Spectra from Proteomic Experiments: Detection, Significance and Utility. *Anal. Chem.*, 75(10), 2003.

[34] Rui Xu and Donald Wunsch. Survey of Clustering Algorithms. *IEEE Transactions on neural networks*, 16(3):645–678, 2005.

[35] Alexander Hinneburg and Daniel A. Keim. An Efficient Approach to Clustering in Large Multimedia Databases with Noise. In *Proc. of KDD'98*, pages 58–65, 1998.

[36] Alexey I. Nesvizhskii. A Survey of Computational Methods and Error Rate Estimation Procedures for Peptide and Protein Identification in Shotgun Proteomics. *Journal of Proteomics*, 73(11):2092–2123, 2010.

[37] Henry Lam and Ruedi Aebersold. Building and Searching Tandem Mass (MS/MS) Spectral Libraries for Peptide Identification in Proteomics. *Methods*, 54(4):424–431, 2011.

[38] R. Craig, J. C. Cortens, D. Fenyo, and R. C. Beavis. Using Annotated Peptide Mass Spectrum Libraries for Protein Identification. *Journal of Proteome Research*, 5(8):1843–1849, 2006.

[39] Henry Lam, Eric W. Deutsch, James S. Eddes, Jimmy K. Eng, Nichole King, Stephen E. Stein, and Ruedi Aebersold. Development and Validation of a Spectral Library Searching Method for Peptide Identification from MS/MS. *Proteomics*, 7(5):655–667, 2007.

[40] Barbara E. Frewen, Gennifer E. Merrihew, Christine C. Wu, William Stafford Noble, and Michael J. MacCoss. Analysis of Peptide MS/MS Spectra from Large-Scale Proteomics Experiments Using Spectrum Libraries. *Analytical Chemistry*, 78(16):5678–5684, 2006.

[41] William R. Cannon, Mitchell M. Rawlins, Douglas J. Baxter, Stephen J. Callister, Mary S. Lipton, and Donald A. Bryant. Large Improvements in MS/MS-Based Peptide Identification Rates using a Hybrid Analysis. *Journal of Proteome Research*, 10(5):2306–2317, 2011.

[42] Attila Kertesz-Farkas, Beata Reiz, Michael P. Myers, and Sandor Pongor. Database Searching in Mass Spectrometry Based Proteomics. *Current Bioinformatics*, 7(2):221–230, 2012.

[43] Michael Kinter and Nicholas E. Sherman. *Protein Sequencing and Identification Using Tandem Mass Spectrometry*. John Wiley & Sons, New York, USA, 2000.

[44] Rovshan G. Sadygov, Daniel Cociorva, and John R. Yates III. Large-scale Database Searching Using Tandem Mass Spectra: Looking up the Answer in the Back of the Book. *Nature Met.*, 1(3):195–202, 2004.

[45] David L. Tabb, Christopher G. Fernando, and Matthew C. Chambers. MyriMatch: Highly Accurate Tandem Mass Spectral Peptide Identification by Multivariate Hypergeometric Analysis. *Journal of Proteome Research*, 6(2):654–661, 2007.

[46] ProteinProspector. `http://prospector.ucsf.edu/`.

[47] Craig D. Wenger and Joshua J. Coon. A Proteomics Search Algorithm Specifically Designed for High-Resolution Tandem Mass Spectra. *Journal of Proteome Research*, 12(3):1377–1386, 2013.

[48] Mass Spectrometry Software. `http://en.wikipedia.org/wiki/Mass_spectrometry_software`.

[49] Neil C. Jones and Pavel A. Pevzner. *An Introduction to Bioinformatics Algorithms*. MIT Press, Cambridge, Massachusetts, 2004.

[50] Stephen Tanner and et al. InsPecT: Identification of Posttranslationally Modified Peptides from Tandem Mass Spectra. *Analytical Chemistry*, 77(14):4626–4639, 2005.

[51] Dekel Tsur, Stephen Tanner, Ebrahim Zandi, Vineet Bafna, and Pavel A. Pevzner. Identification of Post-translational Modifications by Blind Search of Mass Spectra. In *Nature Biotechnology*, volume 23, pages 1562–1567, 2005.

[52] Kang Ning, Hoong K. Ng, and Hon W. Leong. An Accurate and Efficient Algorithm for Peptide and PTM Identification by Tandem Mass Spectrometry. In *Genome Informatics*, volume 19, pages 119–130, 2007.

[53] Jian Liu and et al. Methods for Peptide Identification by Spectral Comparison. *Proteome Science*, 5(3), 2007.

[54] Sven Nahnsen, Timo Sachsenberg, and Oliver Kohlbacher. PTMeta: Increasing Identification Rates of Modified Peptides using Modification Prescanning and Meta-analysis. *Proteomics*, 13(6):1042–1051, 2013.

[55] Jimmy Eng, Ashley L. McCormack, and John R. Yates III. An Approach to Correlate Tandem Mass Spectral Data of Peptides with Amino Acid Sequences in a Protein Database. *J. of the Am. Soc. for Mass Spec.*, 5:976–989, 1994.
`http://fields.scripps.edu/sequest/`.

[56] Andrew Keller, Alexey I. Nesvizhskii, Eugene Kolker, and Ruedi Aebersold. Empirical Statistical Model To Estimate the Accuracy of Peptide Identifications Made by MS/MS and Database Search. *Analytical Chemistry*, 74(20):5383–5392, 2002.
`http://peptideprophet.sourceforge.net/`.

[57] Andrew Keller and et al. Experimental Protein Mixture for Validating Tandem Mass Spectral Analysis. *OMICS: A Journal of Integrative Biology*, 6(2):207–212, 2002.

[58] Christopher Y. Park and et al. Rapid and Accurate Peptide Identification from Tandem Mass Spectra. *Journal of Proteome research*, 7(7):3022–3027, 2008.

[59] Benjamin J. Diament and William S. Noble. Faster SEQUEST Searching for Peptide Identification from Tandem Mass Spectra. *Journal of Proteome Research*, 10(9):3871–3879, 2011.

[60] David N. Perkins, Darryl J. C. Pappin, David M. Creasy, and John S. Cottrell. Probability-based Protein Identification by Searching Sequence Databases using Mass Spectrometry Data. *Electrophoresis*, 20(18):3551–3567, 1999.
`http://www.matrixscience.com/`.

[61] Lewis Y. Geer and et al. Open Mass Spectrometry Search Algorithm. *Journal of Proteome Research*, 3(5):958–964, 2004.

[62] Robertson Craig and Ronald C. Beavis. TANDEM: Matching Proteins with Tandem Mass Spectra. *Bioinformatics*, 20(9):1466–1467, 2004.
`http://www.thegpm.org/TANDEM/`.

[63] Vlado Dančík and et al. De Novo Peptide Sequencing via Tandem Mass Spectrometry. *Journal of Computational Biology*, 6(3):327–342, 1999.

[64] Sangtae Kim, Nuno Bandeira, and Pavel A. Pevzner. Spectral Profiles, a Novel Representation of Tandem Mass Spectra and Their Applications for de Novo Peptide Sequencing and Identification. *Molecular and Cellular Proteomics*, 8(6):1391–1400, 2009.

[65] Bin Ma and et al. PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry. *Rapid Communications in Mass Spectrometry*, 17(20):2337–2342, 2003.

[66] Ari Frank and Pavel Pevzner. PepNovo: De Novo Peptide Sequencing via Probabilistic Network Modeling. *Analytical Chemistry*, 77(4):964–973, 2005.

[67] Richard S. Johnson and J. Alex Taylor. Searching Sequence Databases via De Novo Peptide Sequencing by Tandem Mass Spectrometry. *Molecular Biotechnology*, 22:301–315, 2002.

[68] Marshall Bern, Yuhan Cai, and David Goldberg. Lookup Peaks: A Hybrid of de Novo Sequencing and Database Search for Protein Identification by Tandem Mass Spectrometry. *Analytical Chemistry*, 79(4):1393–1400, 2007.

[69] Lukas Käll and et al. Assigning Significance to Peptides Identified by Tandem Mass Spectrometry Using Decoy Databases. *Journal of Proteome Research*, 7(1):29–34, 2008.

[70] Joshua E. Elias and Steven P. Gygi. Target-Decoy Search Strategy for Mass Spectrometry-Based Proteomics. In *Proteome Bioinformatics*, volume 604, pages 55–71. 2010.

[71] Lukas Käll, John D. Storey, Michael J. MacCoss, and William Stafford Noble. Posterior Error Probabilities and False Discovery Rates: Two Sides of the Same Coin. *Journal of Proteome Research*, 7(1):40–44, 2008.

[72] Sven Nahnsen, Andreas Bertsch, Jörg Rahnenführer, Alfred Nordheim, and Oliver Kohlbacher. Probabilistic Consensus Scoring Improves Tandem Mass Spectrometry Peptide Identification. *Journal of Proteome Research*, 10(8):3332–3343, 2011.

[73] Brian C. Searle, Mark Turner, and Alexey I. Nesvizhskii. Improving Sensitivity by Probabilistically Combining Results from Multiple MS/MS Search Methodologies. *Journal of Proteome Research*, 7(1):245–253, 2008. `http://www.proteomesoftware.com`.

[74] Taejoon Kwon, Hyungwon Choi, Christine Vogel, Alexey I. Nesvizhskii, and Edward M. Marcotte. MSblender: A Probabilistic Approach for Integrating Peptide Identifications from Multiple Database Search Engines. *Journal of Proteome Research*, 10(7):2949–2958, 2011.

[75] David Shteynberg and et al. iProphet: Multi-level Integrative Analysis of Shotgun Proteomic Data Improves Peptide and Protein Identification Rates and Error Estimates. *Molecular and Cellular Proteomics*, 10(12), 2011.

[76] Nathan Edwards, Xue Wu, and Chau-Wen Tseng. An Unsupervised, Model-Free, Machine-Learning Combiner for Peptide Identifications from Tandem Mass Spectra. *Clinical Proteomics*, 5:23–36, 2009.

[77] Neil L. Kelleher, Hong Y. Lin, Gary A. Valaskovic, David J. Aaserud, Einar K. Fridriksson, and Fred W. McLafferty. Top Down versus Bottom Up Protein Characterization by Tandem High-Resolution Mass Spectrometry. *Journal of the American Chemical Society*, 121(4):806–812, 1999.

[78] Alexey I. Nesvizhskii and Ruedi Aebersold. Interpretation of Shotgun Proteomic Data: The Protein Inference Problem. *Molecular and Cellular Proteomics*, 4(10):1419–1440, 2005.

[79] Alexey I. Nesvizhskii, Andrew Keller, Eugene Kolker, and Ruedi Aebersold. A Statistical Model for Identifying Proteins by Tandem Mass Spectrometry. *Analytical Chemistry*, 75(17):4646–4658, 2003. `http://proteinprophet.sourceforge.net`.

[80] Lukas Reiter and et al. Protein Identification False Discovery Rates for Very Large Proteomics Data Sets Generated by Tandem Mass Spectrometry. *Molecular and Cellular Proteomics*, 8(11):2405–2417, 2009.

[81] Manfred Claassen, Lukas Reiter, Michael O. Hengartner, Joachim M. Buhmann, and Ruedi Aebersold. Generic Comparison of Protein Inference Engines. *Molecular and Cellular Proteomics*, 2011.

[82] Gavin E. Reid and Scott A. McLuckey. 'Top Down' Protein Characterization via Tandem Mass Spectrometry. *Journal of Mass Spectrometry*, 37(7):663–675, 2002.

[83] Marcus Bantscheff, Markus Schirle, Gavain Sweetman, Jens Rick, and Bernhard Kuster. Quantitative Mass Spectrometry in Proteomics: a Critical Review. *Analytical and Bioanalytical Chemistry*, 389:1017–1031, 2007.

[84] Marcus Bantscheff, Simone Lemeer, MikhailM. Savitski, and Bernhard Kuster. Quantitative Mass Spectrometry in Proteomics: Critical Review Update from 2007 to the Present. *Analytical and Bioanalytical Chemistry*, 404:939–965, 2012.

[85] Jeffrey C. Silva, Marc V. Gorenstein, Guo-Zhong Li, Johannes P. C. Vissers, and Scott J. Geromanos. Absolute Quantification of Proteins by LCMSE: A Virtue of Parallel ms Acquisition. *Molecular and Cellular Proteomics*, 5(1):144–156, 2006.

[86] Clemens Gröpl and et al. Algorithms for the Automated Absolute Quantification of Diagnostic Markers in Complex Proteomics Samples. In *Proceedings of the First international conference on Computational Life Sciences*, CompLife'05, pages 151–162, 2005.

[87] Steven P. Gygi, Beate Rist, Scott A. Gerber, Frantisek Turecek, Michael H. Gelb, and Ruedi Aebersold. Quantitative Analysis of Complex Protein Mixtures using Isotope-coded Affinity Tags. *Nat Biotech*, 17:994–999, 1999.

[88] Philip L. Ross, Yulin N. Huang, and et al. Multiplexed Protein Quantitation in Saccharomyces cerevisiae Using Amine-reactive Isobaric Tagging Reagents. *Molecular and Cellular Proteomics*, 3(12):1154–1169, 2004.

[89] Ian I. Stewart, T. Thomson, and D. Figeys. ^{18}O Labeling: a Tool for Proteomics. *Rapid Communications in Mass Spectrometry*, 15(24):2456–2465, 2001.

[90] Shao-En Ong, Blagoy Blagoev, Irina Kratchmarova, Dan Bach Kristensen, Hanno Steen, Akhilesh Pandey, and Matthias Mann. Stable Isotope Labeling by Amino Acids in Cell Culture, SILAC, as a Simple and Accurate Approach to Expression Proteomics. *Molecular and Cellular Proteomics*, 1(5):376–386, 2002.

[91] SILAC on Wikipedia. `http://en.wikipedia.org/wiki/Stable_isotope_labeling_by_amino_acids_in_cell_culture`.

[92] Jurgen Cox and Matthias Mann. MaxQuant Enables High Peptide Identification Rates, Individualized p.p.b.-range Mass Accuracies and Proteome-wide Protein Quantification. *Nat Biotech*, 26:1367–1372, 2008.

[93] Marc Sturm and et al. OpenMS – An Open-source Software Framework for Mass Spectrometry. *BMC Bioinformatics*, 9(163), 2008. `http://www.open-ms.de/`.

[94] Matthew C. Wiener, Jeffrey R. Sachs, Ekaterina G. Deyanova, and Nathan A. Yates. Differential Mass Spectrometry: A Label-Free LC-MS Method for Finding Significant Differences in Complex Peptide and Protein Mixtures. *Analytical Chemistry*, 76(20):6085–6096, 2004.

[95] Weixun Wang, Haihong Zhou, Hua Lin, Sushmita Roy, Thomas A. Shaler, Lander R. Hill, Scott Norton, Praveen Kumar, Markus Anderle, and Christopher H. Becker. Quantification of Proteins and Metabolites by Mass Spectrometry without Isotopic Labeling or Spiked Standards. *Analytical Chemistry*, 75(18):4818–4826, 2003.

[96] Sven Nahnsen, Chris Bielow, Knut Reinert, and Oliver Kohlbacher. Tools for Label-free Peptide Quantification. *Molecular and Cellular Proteomics*, 2012.

[97] Deborah H. Lundgren, Sun-Il Hwang, Linfeng Wu, and David K. Han. Role of Spectral Counting in Quantitative Proteomics. *Expert Review of Proteomics*, 7(1):39–53, 2010.

[98] Marc Sturm. *OpenMS – A Framework for Computational Mass Spectrometry*. PhD thesis, Universität Tübingen, 2010.

[99] Valerio B. Di Marco and G. Giorgio Bombi. Mathematical Functions for the Representation of Chromatographic Peaks. *Journal of Chromatography A*, 931(1–2):1–30, 2001.

[100] Michael W. Senko, Steven C. Beu, and Fred W. McLaffertycor. Determination of Monoisotopic Masses and Ion Populations for Large Biomolecules from Resolved Isotopic Distributions. *Journal of the American Society for Mass Spectrometry*, 6:229–233, 1995.

[101] Eva Lange, Clemens Gröpl, Ole Schulz-Trieglaff, Andreas Leinenbach, Christian Huber, and Knut Reinert. A Geometric Approach for the Alignment of Liquid Chromatography – Mass Spectrometry Data. *Bioinformatics*, 23(13):i273–i281, 2007.

[102] Katharina Podwojski and et al. Retention Time Alignment Algorithms for LC/MS Data Must Consider Non-linear Shifts. *Bioinformatics*, 25(6):758–764, 2009.

[103] Eric W. Deutsch and et al. A guided tour of the Trans-Proteomic Pipeline. *Proteomics*, 10(6):1150–1159, 2010.

[104] Oliver Kohlbacher and et al. TOPP – the OpenMS Proteomics Pipeline. *Bioinformatics*, 23(2):e191–e197, 2007.

[105] Johannes Junker, Chris Bielow, Andreas Bertsch, Marc Sturm, Knut Reinert, and Oliver Kohlbacher. TOPPAS: A Graphical Workflow Editor for the Analysis of High-Throughput Proteomics Data. *Journal of Proteome Research*, 11(7):3914–3920, 2012.

[106] Marc Sturm and Oliver Kohlbacher. TOPPView: An Open-Source Viewer for Mass Spectrometry Data. *Journal of Proteome Research*, 8(7):3760–3763, 2009.

[107] UniProtKB/Swiss-Prot Statistics. `http://web.expasy.org/docs/relnotes/relstat.html`.

[108] R. Bayer and E. M. McCreight. Organization and Maintenance of Large Ordered Indices. In *Acta Inf.*, volume 1, pages 173–189, 1972.

[109] Karl R. Clauser, Peter Baker, and Alma L. Burlingame. Role of Accurate Mass Measurement (±10 ppm) in Protein Identification Strategies Employing MS or MS/MS and Database Searching. *Analytical Chemistry*, 71(14):2871–2882, 1999.

[110] Jesper V. Olsen and et al. Parts per Million Mass Accuracy on an Orbitrap Mass Spectrometer via Lock Mass Injection into a C-trap. *Molecular and Cellular Proteomics*, 4(12):2010–2021, 2005.

[111] Ejvind Mørtz and et al. Sequence Tag Identification of Intact Proteins by Matching Tandem Mass Spectral Data Against Sequence Databases. In *Proc. Nati. Acad. Sci. USA*, volume 93, pages 8264–8267, 1996.

[112] M. Mann and M. Wilm. Error-tolerant Identification of Peptides in Sequence Databases by Peptide Sequence Tags. *Analytical chemistry*, 66:4390–4399, 1994.

[113] David L. Tabb, Anita Saraf, and John R. Yates III. GutenTag: High-Throughput Sequence Tagging via an Empirically Derived Fragmentation Model. *Analytical Chemistry*, 75(23):6415–6421, 2003.

[114] Smriti R. Ramakrishnan and et al. A Fast Coarse Filtering Method for Peptide Identification by Mass Spectrometry. *Bioinformatics*, 22(12):1524–1531, 2006.

[115] Daniel Miranker, Weijia Xu, and Rui Mao. MoBIoS: a Metric-Space DBMS to Support Biological Discovery. In *Proc. of the Int. Conf. on Scientific and Statistical Database Management Systems*, pages 241–244, 2003.

[116] Pavel Zezula, Giuseppe Amato, Vlastislav Dohnal, and Michal Batko. *Similarity Search: The Metric Space Approach (Advances in Database Systems)*. Springer, USA, 2006.

[117] Tolga Bozkaya and Meral Ozsoyoglu. Distance-based Indexing for High-dimensional Metric Spaces. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 357–368, 1997.

[118] S. Cenk Sahinalp, Murat Tasan, Jai Macker, and Z. Meral Ozsoyoglu. Distance Based Indexing for String Proximity Search. In *Proc. of 19th International Conference on Data Engineering*, pages 125–136, 2003.

[119] Debojyoti Dutta and Ting Chen. Speeding up Tandem Mass Spectrometry Database Search: Metric Embeddings and Fast Near Neighbor Search. *Bioinformatics*, 23(5):612–618, 2007.

[120] Søren Besenbacher, Benno Schwikowski, and Jens Stoye. Indexing and Searching a Mass Spectrometry Database. In *Algorithms and Applications*, volume 6060 of *Lecture Notes in Computer Science*, pages 62–76. 2010.

[121] Mohamed Helmy, Masaru Tomita, and Yasushi Ishihama. Peptide Identification by Searching Large-scale Tandem Mass Spectra against Large Databases: Bioinformatics Methods in Proteogenomics. *Genes Genome Genomics*, 6:76–85, 2012.

[122] Alexandr Andoni and Piotr Indyk. Near-optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[123] Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. Locality Sensitive Hashing: a Comparison of Hash Function Types and Querying Mechanisms. *Pattern Recognition Letters*, 31(11):1348–1358, 2010.

[124] David Novak, Martin Kyselak, and Pavel Zezula. On Locality-sensitive Indexing in Generic Metric Spaces. In *SISAP '10*, pages 67–74, 2010.

[125] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[126] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive Hashing Scheme Based on P-stable Distributions. In *SCG '04*, pages 253–262, 2004.

[127] Rui Mao, Smriti R. Ramakrishnan, Glen Nuckolls, and Daniel P. Miranker. An Inverted Index for Mass Spectra Similarity Query and Comparison with a Metric-space Method: Case Study. In *SISAP '10*, pages 93–99, 2010.

[128] Houjun Tang and et al. An Inverted Index Method for Mass Spectra K-Nearest Neighbor Queries. *Lecture Notes in Information Technology*, 10:115–122, 2012.

[129] You Li and et al. Speeding up Tandem Mass Spectrometry Based Database Searching by Peptide and Spectrum Indexing. *Rapid Comm. Mass Spec.*, 24(6):807–814, 2010.

[130] Xiaowen Liu, Alessandro Mammana, and Vineet Bafna. Speeding up Tandem Mass Spectral Identification using Indexes. *Bioinformatics*, 28(13):1692–1697, 2012.

[131] Zhan Wu, Gilles Lajoie, and Bin Ma. MSDASH: Mass Spectrometry Database and Search. *Proc. LSS Comput. Syst. Bioinform. Conf.*, 7:63–71, 2008.

[132] Peter J. Ulintz, Ji Zhu, Zhaohui S. Qin, and Philip C. Andrews. Improved Classification of Mass Spectrometry Database Search Results Using Newer Machine Learning Approaches. *Molecular and Cellular Proteomics*, 5:497–509, 2006.

[133] D. C. Anderson, Weiqun Li, Donald G. Payan, and William S. Noble. A New Algorithm for the Evaluation of Shotgun Peptide Sequencing in Proteomics: Support Vector Machine Classification of Peptide MS/MS Spectra and SEQUEST Scores. *Journal of Proteome Research*, 2(2):137–146, 2003.

[134] Kang Ning, Hoong K. Ng, and Hon W. Leong. PepSOM: An Algorithm for Peptide Identification by Tandem Mass Spectrometry based on SOM. In *Genome Informatics*, volume 17, pages 194–205, 2006.

[135] Bingwen Lu and Ting Chen. A Suffix Tree Approach to the Interpretation of Tandem Mass Spectra: Applications to Peptides of Non-specific Digestion and Post-translational Modifications. In *Bioinformatics*, volume 19, pages Suppl. 2:ii113–21, 2003.

[136] Esko Ukkonen. On-line Construction of Suffix Trees. *Algorithmica*, 14:249–260, 1995.

[137] Chen Zhou and et al. Speeding up Tandem Mass Spectrometry-based Database Searching by Longest Common Prefix. *BMC Bioinformatics*, 11:1–11, 2010.

[138] Leheng Wang and et al. An Efficient Parallelization of Phosphorylated Peptide and Protein Identification. *Rapid Communications in Mass Spectrometry*, 24(12):1791–1798, 2010.

[139] You Li and Xiaowen Chu. Speeding up Scoring Module of Mass Spectrometry Based Protein Identification by GPU. In *2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, pages 1315–1320, 2012.

[140] Istvan Bogdan, Daniel Coca, Jenny Rivers, and Robert J. Beynon. Hardware Acceleration of Processing of Mass Spectrometric Data for Proteomics. *Bioinformatics*, 23(6):724–731, 2007.

[141] Tomáš Skopal. Unified Framework for Fast Exact and Approximate Search in Dissimilarity Spaces. *ACM Transactions on Database Systems*, 32(4):29, 2007.

[142] María Luisa Micó, José Oncina, and Enrique Vidal. A New Version of the Nearest-neighbour Approximating and Eliminating Search Algorithm (AESA) with Linear Preprocessing Time and Memory requirements. *Pattern Recogn. Lett.*, 15(1):9–17, 1994.

[143] Francisco Moreno-Seco, Luisa Micó, and Jose Oncina. Extending LAESA Fast Nearest Neighbour Algorithm to Find the k Nearest Neighbours. *Structural, Syntactic, and Statistical Pattern Recognition*, LNCS 2396:718–724, 2002.

[144] Enrique Vidal. New Formulation and Improvements of the Nearest-neighbour Approximating and Eliminating Search Algorithm (AESA). *Pattern Recogn. Lett.*, 15(1):1–7, 1994.

[145] Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB*, pages 426–435, 1997.

[146] Marco Patella. *Similarity Search in Multimedia Databases*. Dipartimento di Elettronica Informatica e Sistemistica, Bologna, 1999. `http://www-db.deis.unibo.it/Mtree/index.html`.

[147] Jakub Lokoč. Parallel Dynamic Batch Loading in the M-tree. In *SISAP 2009, IEEE*, pages 117–123, 2009.

[148] Tomáš Skopal, Jaroslav Pokorný, and Václav Snášel. PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. In *ADBIS*, pages 99–114, 2004.

[149] Jakub Lokoč, Přemysl Čech, Jiří Novák, and Tomáš Skopal. Cut-Region: A Compact Building Block for Hierarchical Metric Indexing. *Similarity Search and Applications*, LNCS 7404:85–100, 2012.

[150] Edgar Chávez and Gonzalo Navarro. A Probabilistic Spell for the Curse of Dimensionality. In *ALENEX'01, LNCS 2153*, pages 147–160. Springer, 2001.

[151] Tomáš Skopal and Benjamin Bustos. On Nonmetric Similarity Search Problems in Complex Domains. *ACM Computing Surveys*, 43(4), 2011.

[152] Tomáš Skopal and Jakub Lokoč. NM-Tree: Flexible Approximate Similarity Search in Metric and Non-metric Spaces. In *DEXA '08*, pages 312–325, 2008.

[153] Jiří Novák, Tomáš Skopal, David Hoksza, and Jakub Lokoč. Non-metric Similarity Search of Tandem Mass Spectra Including Posttranslational Modifications. *Journal of Discrete Algorithms*, 13:19–31, 2012.

[154] Jiří Novák, Tomáš Skopal, David Hoksza, and Jakub Lokoč. Improving the Similarity Search of Tandem Mass Spectra using Metric Access Methods. In *SISAP '10*, pages 85–92, 2010.

[155] Jiří Novák, David Hoksza, Jakub Lokoč, and Tomáš Skopal. On Optimizing the Non-metric Similarity Search in Tandem Mass Spectra by Clustering. *Bioinformatics Research and Applications*, LNBI 7292:189–200, 2012.

[156] Katty X. Wan, Ilan Vidavsky, and Michael L. Gross. Comparing Similar Spectra: From Similarity Index to Spectral Contrast Angle. *Journal of the American Society for Mass Spectrometry*, 13(1):85–88, 2002.

[157] Z. B. Alfassi. On the Normalization of a Mass Spectrum for Comparison of Two Spectra. *Journal of the Am. Soc. for Mass Spec.*, 15(3):385–387, 2004.

[158] Jiří Novák and David Hoksza. An Application of the Metric Access Methods to the Mass Spectrometry Data. In *CIBCB 2009, IEEE*, pages 220–227, 2009.

[159] Jiří Novák and David Hoksza. Parametrised Hausdorff Distance as a Non-Metric Similarity Model for Tandem Mass Spectrometry. In *CEUR Proc. DATESO*, pages 1–12, 2010.

[160] Jiří Novák, Timo Sachsenberg, David Hoksza, Tomáš Skopal, and Oliver Kohlbacher. A Statistical Comparison of SimTandem with State-of-the-Art Peptide Identification Tools. In *7th International Conference on Practical Applications of Computational Biology and Bioinformatics*, accepted, 2013.

[161] Jiří Novák and David Hoksza. Similarity Search and Posttranslational Modifications in Tandem Mass Spectra. In *IEEE Bioinformatics and Biomedicine Workshops*, pages 845–846, 2010.

[162] Jian Wang and et al. Peptide Identification from Mixture Tandem Mass Spectra. *Molecular and Cellular Proteomics*, 9(7):1476–1485, 2010.

[163] Jiří Novák, Tomáš Skopal, David Hoksza, Jakub Lokoč, and Jakub Galgonek. Protein Sequences Identification using NM-tree. In *SISAP '11*, pages 125–126, 2011.

[164] The Global Proteome Machine Organization (GPM). `http://www.thegpm.org/quartz/`.

[165] Martin Beck and et al. The Quantitative Proteome of a Human Cell Line. *Molecular Systems Biology*, 7(549), 2011.

[166] Jiří Novák, Jakub Galgonek, David Hoksza, and Tomáš Skopal. SimTandem: Similarity Search in Tandem Mass Spectra. *Similarity Search and Applications*, LNCS 7404:242–243, 2012.
`http://www.simtandem.org`,
`http://siret.cz/simtandem`.

[167] E. coli and Human Data Sets.
E. coli
http://www-bs2.informatik.uni-tuebingen.de/services/sachsenb/data/20100219_SvNa_SA_Ecoli.mzML,
Hum48
http://www-bs2.informatik.uni-tuebingen.de/services/sachsenb/data/A10-07048.mzXML,
Hum49
http://www-bs2.informatik.uni-tuebingen.de/services/sachsenb/data/A10-07049.mzXML.

[168] SImilarity RETrieval Research Group (SIRET).
http://www.siret.cz/.

[169] Threading Building Blocks.
http://www.threadingbuildingblocks.org/.

[170] Google Web Toolkit (GWT).
http://developers.google.com/web-toolkit/.

[171] omniORB : Free CORBA ORB.
http://omniorb.sourceforge.net/.

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AA-residue	Amino Acid Residue
AESA	Approximating and Eliminating Search Algorithm
ANN	Approximate Nearest Neighbor
Da	Dalton
DDH	Distance Distribution Histogram
DNA	Deoxyribonucleic Acid
ESI	Electrospray Ionization
FDR	False Discovery Rate
FP-modifier	Fractional-Power Modifier
GPU	Graphic Processing Unit
HDD	Hard Disk Drive
HPLC	High-pressure Liquid Chromatography
ICAT	Isotope-coded Affinity Tags
iTRAQ	Isobaric Tags for Relative an Absolute Quantification
kNN	k Nearest Neighbors
LAESA	Linear Approximating and Eliminating Search Algorithm
LBQ	Label-based Quantification
LFQ	Label-free Quantification
LSH	Locality Sensitive Hashing
M-tree	Metric Tree
MALDI	Matrix Assisted Laser Desorption Ionization
MAM	Metric Access Method
MOWSE	MOlecular Weight SEarch
MS	Mass Spectrometry
MS/MS, MS^2	Tandem Mass Spectrometry
MSDB	Mass Spectrometry Protein Sequence Database
MVP-tree	Multi Vantage Point Tree
NAM	Non-metric Access Method
NM-tree	Non-metric Tree
NSP	Number of Sibling Peptides
OMSSA	Open Mass Spectrometry Search Algorithm
PEP	Posterior Error Probability
PM-tree	Pivoting Metric Tree
PPE	Peptide Probability Estimate
ppm	Parts per Million

PSM	Peptide-spectrum Match
PST	Peptide Sequence Tag
PTM	Post-translational Modification
RBQ-modifier	Rational-Bézier-Quadratic Modifier
RNA	Ribonucleic Acid
SILAC	Stable Isotope Labeling by Amino Acids in Cell Culture
SOM	Self-organizing Map
SP-modifier	Similarity Preserving Modifier
SPC	Shared Peak Count
SSD	Solid-state Drive
T-error	Triangle Error
T-modifier	Triangle Modifier
TG-modifier	Triangle Generating Modifier
TOF	Time of Flight
TOPP	The OpenMS Proteomics Pipeline
TOPPAS	The OpenMS Proteomics Pipeline ASsistant
TPP	Trans-Proteomic Pipeline
TriGen	Triangle Generating Algorithm
TV-modifier	Triangle Violating Modifier
VP-tree	Vantage Point Tree