New Statistical Algorithms for the Analysis of Mass Spectrometry Time-Of-Flight Mass Data with Applications in Clinical Diagnostics

by

Tim OF Conrad, BSc, BCompSc (Hon), MSc

Thesis

Submitted by Tim OF Conrad

for fulfillment of the Requirements for the Degree of

Doktor der Naturwissenschaften

Supervisor: Professor Dr. Christof Schütte (FUB)

Associate Supervisor: Professor Dr. Knut Reinert (FUB)

Reviewer: Dr. André Hagehülsmann (MSR)

Disputation: 10.07.2008

Department of Mathematics and Computer Science
Freie Universität Berlin

May, 2008

Contents

New Statistical Algorithms for the Analysis of Mass Spectrometry Time-Of-Flight Mass Data with Applications in Clinical Diagnostics

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Tim OF Conrad
September 30, 2008

Acknowledgments

Diese zum Schluss verfassten Zeilen meiner Dissertation möchte ich nutzen, um denjenigen zu danken, die mich in den letzten drei Jahren begleitet, unterstützt und gefördert haben.

Mein besonderer Dank gilt meinem Betreuer Prof. Christof Schütte für die ständige Unterstützung und Förderung, vor allem aber für den gewährten Freiraum, um eigene Ideen entwickeln zu können. Weiterhin möchte ich mich auch bei meinem Zweitbetreuer Prof. Knut Reinert für wertvolle Hinweise und Anregungen und bei Dr. André Hagehülsmann für die Übernahme des Koreferats und seiner Hilfe bei der Einwerbung der Projektfinanzierung bedanken.

Ein herzlicher Dank gilt meinen Eltern, meinen Brüdern Kay und Jens und meinen Freunden und Kollegen innerhalb und außerhalb der BioComputing Arbeitsgruppe - insbesondere Jana -, die alle auf ihre Weise zum Gelingen dieser Arbeit beigetragen haben.

Zum Schluss möchte ich noch der Firma Microsoft Research Ltd. in Cambridge (UK) danken, die mich durch ein Promotionsstipendium finanziell unterstützt hat.

Tim OF Conrad

Freie Universität Berlin
May 2008

Extended Abstract

English Version

Mass spectrometry (MS) based techniques have emerged as a standard for large-scale protein analysis. The ongoing progress in terms of more sensitive machines and improved data analysis algorithms led to a constant expansion of its fields of applications. Recently, MS was introduced into clinical proteomics with the prospect of early disease detection using proteomic pattern matching.

MS standard for large scale protein analysis

MS for early disease detection

Analyzing biological samples (e.g. blood) by mass spectrometry generates *mass spectra* that represent the components (molecules) contained in a sample as masses and their respective relative concentrations. It is well known that an individual's proteome is highly dynamic and changes quite dramatically during a day, depending on a variety of factors. However, analyzing a large enough group of similar individuals (e.g. "healthy" or "suffering from disease X") allows to identify components in the respective spectra that do not differ much - with respect to concentration - between individuals from the same group (*constant* components).

Mass spectra represent relative concentrations of molecules in a sample

An individual's proteome is highly dynamic and influenced by e.g. diseases

In this work, we are interested in those components that are constant within a group of individuals but differ much between individuals of two distinct groups. These distinguishing components that dependent on a particular medical condition are generally called *biomarkers.* Since not all biomarkers found by the algorithms are of equal (discriminating) quality we are only interested in a small biomarker subset that - as a combination - can be used as a *fingerprint* for a disease. Once a fingerprint for a particular disease (or medical condition) is identified, it can be used in clinical diagnostics to classify unknown spectra.

In this thesis: search for spectra components that reflect particular diseases

Best components can be combined to a disease's fingerprint

This mass spectrometry based method appears to be one of the arising key technologies for biomarker discovery, understanding of biological mechanisms, and consequently, it might offer new approaches in drug development.

In this thesis we have developed new algorithms for automatic extraction of disease specific fingerprints from mass spectrometry data. Special emphasis has been put on designing highly sensitive methods with respect to signal detection. This is extremely important in all stages of the pipeline (such as spectra preprocessing, signal detection, signal analysis and identification of disease specific fingerprints) since many biologically relevant molecules are found to be very low abundant (such as hormones) thus yielding (comparatively) small signals. Thanks to our statistically based approach our methods are able to detect signals even below the noise level inherent in data acquired by common MS machines.

New algorithms for automatic extraction of fingerprints have been developed

Focus was laid on developing highly sensitive algorithms to allow detection of low abundant molecules within noise

To provide access to these new classes of algorithms to collaborating groups we have created a web-based analysis platform that provides all necessary interfaces for data transfer, data analysis and result inspection. Following

A new analysis platform for signal (pre-)processing and extraction of disease specific fingerprints was developed

fingerprint extraction, the platform provides efficient and robust classification algorithms to determine the medical condition of an individual.

Platform has already been used in clinical studies

To prove the platform's practical relevance it has been utilized in several clinical studies two of which are presented in this thesis. In these studies it could be shown that our platform is superior to commercial systems with respect to fingerprint identification. As an outcome of these studies several fingerprints for different cancer types (bladder, kidney, testicle, pancreas, colon and thyroid) have been detected and validated. The clinical partners in fact emphasize that these results would be impossible with a less sensitive analysis tool (such as the currently available systems).

A new ad-hoc Grid infrastructure was developed to cope with the large datasets

In addition to the issue of reliably finding and handling signals in noise we faced the problem to handle very large amounts of data, since an average dataset of an individual is about 2.5 Gigabytes in size and we have data of hundreds to thousands of persons. To cope with these large datasets, we developed a new framework for a heterogeneous (quasi) ad-hoc Grid - an infrastructure that allows to integrate thousands of computing resources (e.g. Desktop Computers, Computing Clusters or specialized hardware, such as IBM's Cell Processor in a Playstation 3).

German Version

Das Gebiet der *Proteomik* (englisch: *proteomics*) umfasst die Erforschung des *Proteoms*, d.h. der Gesamtheit aller in einem Organismus (z.B. Mensch) vorhandenen Proteine. Im Gegensatz zum eher statischen *Genom* ist das Proteom hoch dynamisch, d.h. die Zusammensetzung wie auch die Konzentration der einzelnen Proteine ändert sich über den Tag teilweise dramatisch und wird beeinflusst durch zum Beispiel Umwelteinflüsse, Medikamente oder Krankheiten.

Massenspektrometrie (MS) -basierte Verfahren haben sich als Standardtechnik zur Proteomanalyse etabliert. Diese Verfahren ermöglichen das Bestimmen der (relativen) Konzentrationen von Proteinen in Körperflüssigkeiten, wie zum Beispiel im Blut. In jedem Blutstropfen schwimmt ein vielfältiges Gemisch dieser Eiweiße; Art und Menge variieren von Mensch zu Mensch. Innerhalb dieses Gemisches lassen sich auch Veränderungen entdecken, die durch Krankheiten hervorgerufen werden. Jede Krankheit verändert dabei eine ganz bestimmte Menge von Proteinen (bzw. deren Konzentration) in einer charakteristischen Art und Weise und besitzt damit einen eindeutigen *Fingerabdruck*.

Um einen aussagekräftigen Fingerabdruck für eine bestimmte Krankheit zu finden, müssen zunächst diese Veränderungen (Signale) zwischen den Daten (*Spektren*) von Gesunden und Kranken gefunden werden. Diese Signale repräsentieren Veränderungen in der Konzentration von bestimmten Molekülen zwischen gesunden und kranken Individuen und werden *Biomarker* genannt. Analysen der Unterschiede zwischen einer Gruppe von gesunden und einer Gruppe von kranken Menschen ergeben oft hunderte von verschiedenen Biomarker, die von stark unterschiedlicher Qualität sind (bezogen auf den Unterschied zwischen "gesund" und "krank"). Daher wird für den tatsächlichen Fingerabdruck diejenige Teilmenge aller möglichen Signale benutzt, die sich in Kombination am besten dazu eignen, die beiden Gruppen zu unterscheiden.

Diese gefundenen Fingerabdrücke ermöglichen zum Beispiel die Früherkennung von Krankheiten. Allerdings entstehen hier auch neue Herausforderungen: Zum einen stehen zwar große Datenmengen aus gut charakterisierten Proben für eine valide Statistik zur Verfügung, zum anderen sind gerade die hier benutzten MS Hochdurchsatzverfahren anfällig für Störungen zum Beispiel durch Rauschen und erfordern außerordentlich präzise Algorithmen zur sicheren Signalerkennung innerhalb großer Anzahlen an Datensätzen.

Diese Arbeit beschäftigt sich mit der oben erwähnten Analyse von Daten aus MS-Experimenten und stellt eine neue web-basierte Analyseplattform und neue Verfahren zur Vorverarbeitung, Signalerkennung und Fingerabdruckerkennung vor. Die erreichten Verbesserungen insbesondere der Detektionssensitivität steigern unmittelbar die resultierende Klassifizierungsgüte, über deren zugrundeliegende Signale eine gezielte biochemische Identifikation potentieller Biomarker überhaupt erst möglich wird.

Um die praktische Relevanz der neu entwickelten Algorithmen zu zeigen, wurde die neu entwickelte Plattform bereits in mehreren klinischen Studien eingesetzt - zwei dieser Studien werden in dieser Arbeit ausführlich beschrieben. In diesen Studien wurde gezeigt, dass unsere Verfahren anderen (kommerziellen) Systemen im Bezug auf die Sensitivität bei der Erkennung von Fingerabdrücken überlegen ist. Als Ergebnis dieser Studien wurden neue Fingerabdrücke für verschiedene Krebsarten (u.a. Blase, Niere, Schilddrüse und Bauchspeicheldrüse) gefunden und validiert. Die klinischen Partner haben ausdrücklich betont, dass diese Ergebnisse mit den vorhandenen (weniger sensitiven) Systemen nicht möglich gewesen wären.

Um die Verarbeitung der MS Massendaten zu ermöglichen (ca. 2.5 Gigabyte pro Datensatz bei Tausenden von Datensätzen), wurde ein neues (quasi) ad-hoc Grid System entwickelt. Diese Computerinfrastruktur bietet die Möglichkeit zur Einbindung von Tausenden von Rechenressourcen, zum Beispiel von Desktopcomputern, Computerclustern oder auch Spezialhardware, wie den Cell-Prozessor von IBM in einer Playstation 3.

Chapter 1

Introduction and Survey

1.1 Introduction

The Postgenomic Era

The successful completion of the Human Genome Project, which identified and mapped (almost) the entire collection of human genes (the *genome*), has uncovered an amazing amount of information. However, this information did only build up the foundation for further exploration to ultimately answer the *fundamental question* biologists are interested in: How do the genes in the genome work together in a human being ?

Genome Project: foundation for real question: how do genes work in humans?

Although this and related projects (and genetics in general) have received much attention during the last couple of years one should not forget that the primary purpose of most genes is to code for proteins. Whereas gene structure and organization are important, the fate of a cell (and thus of each organism) is determined by specific proteins and some RNAs (Pandey and Mann, 2000). They execute and control the majority of cellular activities and are the targets of nearly all our drugs. Realizing this, the primary focus of biologists is shifting towards the *proteome*, the set of all proteins the human genome can produce. More specifically, the proteome is the set of expressed proteins *at a given point in time under specific conditions* - and thus dynamic. A snapshot of a cells proteome provides information about the ensemble of proteins active in that cell at that time under the given specific physiological conditions (Wasinger et al., 1995; Wilkins et al., 1996; Naaby-Hansen et al., 2005).

Gene's purpose: code for proteins

Proteins determine fate of a cell

Biologist's focus shifts to proteins

Proteome (= set of all proteins) becomes main topic

Cell proteome provides temporal information about cell state

Unfortunately, switching from genes to proteins adds some orders of magnitude to the complexity of the problem: While the human genome is estimated to contain approximately 30.000-40.000 genes that code for proteins, the corresponding number of proteins these genes encode for is much higher. Events such as alternative splicing of genes and post-translational modifications generate a highly diverse set of proteins that could exceed a million distinct molecular species within a given cell. Most of these being yet uncharacterized (Whisstock and Lesk, 2003).

$3 \cdot 10^4$ genes vs. 10^6 proteins

The Proteomics Era

Bringing proteins into the center of attention, the *fundamental question* from the first paragraph now turns into: What is the specific purpose of each protein, how do they interact with each other, how are they modified and how can we change their actions therapeutically ?

New main question: how do proteins work in humans?

Proteomics (HT technology) = tools for finding answers to main questions

Analytical protein chemistry, or *proteomics* as it is now commonly known, provides the tools for answering these questions - high-throughput technologies for the large-scale, rapid analysis of proteins. Experiments showed an enormous potential in clarifying biochemical and physiological mechanisms of complex diseases at a molecular level (Wittmann and Heinzle, 1999; Süssmuth and Jung, 1999; Qian et al., 2006; Cravatt et al., 2007; Kicman et al., 2007). The Human Proteome Organization (HUPO, 2005) states that

> "The field of proteomics is particularly important because most diseases are manifested at the level of protein activity. Consequently, proteomics seeks to correlate directly the involvement of specific proteins, protein complexes and their modification status in a given disease state. Such knowledge will provide a fast track to commercialization and will speed up the identification of new drug targets that can be used to diagnose and treat diseases."

Interest in applying proteomics for diagnostics

Data analysis is challenging

Need for developing DBs, analyzing and visualizing methods

Data volume is challenging

Motivated by these results, there is intense interest in applying proteomics to foster a better understanding of disease processes, develop new biomarkers for diagnosis and early detection of disease and accelerate drug development. This interest creates numerous opportunities as well as challenges to meet the needs for high sensitivity and high throughput required for disease-related investigations. The handling and analysis of data generated by proteomics investigations represents an emerging and challenging field. New techniques and collaborations between computer scientists, mathematicians and biologists are called for. There is a need to develop and integrate a variety of different types of databases; to develop tools for translating raw primary data into forms suitable for other researchers and formal data analysis; to obtain and develop user interfaces to store, retrieve and visualize the data from databases; and to develop efficient and valid methods of data analysis. The sheer volume of data to be collected and processed will challenge the usual approaches. Analyzing data of this dimension is a fairly new endeavor for all participating scientific fields.

Mass Spectrometry as Dominant Proteomics Technology

MS dominant technology for proteome analysis

Among all proteomic technologies (such as protein microarrays, two-hybrid analyses or crystallization) *mass spectrometry* has emerged as the dominant technique for analyzing production and function of proteins in organisms (Aebersold and Mann, 2003). Simply put[1], a mass spectrum represents a snapshot of the abundances of ions (e.g molecular or fragment ions) contained in a (biological) sample (such as blood serum or other body fluids - see figure 1.1.1 for an example spectrum), plotted against their mass to charge ratio. This is in particular interesting since it allows not only for examining functions of isolated proteins but also to detect molecular modifications (result in modified mass) or monitor changes in concentration.

MS spectra analysis: extract differences

Biomarkers (= molecules different in two patient groups) can indicate disease status

Fingerprints = group of biomarkers: can aid early cancer detection

One way of analyzing mass spectra (which this thesis deals with) is the extraction of significant differences between spectra obtained from different groups of people. For example, spectra from a "healthy" cohort of patients can be compared to those obtained from patients having a particular disease. (Spectra-)Differences between these two groups - which represent differences on the molecular (peptide) level - can then be used as so called *Biomarkers*: indicators for existence, status or progress of a particular disease. Groups of

[1]See section 2.1.1 on page 11 for an introduction to mass spectrometry.

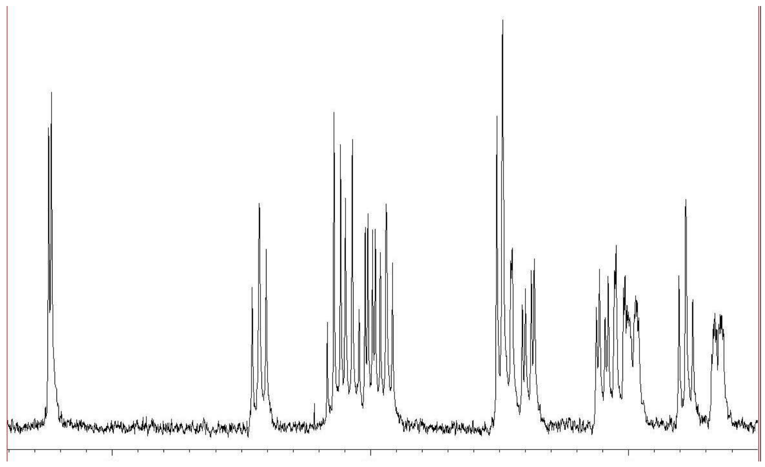


Figure 1.1.1: A small part of a common spectrum. The x axis reflects the mass over charge (m/z) value and the y axis the number of times a particle was counted by the mass spectrometer.

these single biomarkers are called *Fingerprints*: distinct signal patterns representing distinguishing peptide signatures (e.g. protein fragments). Several studies have shown the potential of such patterns for early detection of different types of cancer (see (Kozak et al., 2005; Becker et al., 2004) and our studies presented in chapter 4).

Fingerprints usually hidden and small components hard to detect

Unfortunately, these fingerprints are usually hidden in *much* larger sets of signals, such as other (non distinguishing) peptide signals or noise (Tibshirani et al., 2004; Gillette et al., 2005). Especially small signals - which represent low abundant molecules (such as hormones) - are extremely hard to detect since they are literally buried in noise. In this thesis we will introduce new algorithms to reliably detect even these small signals to allow for much more sensitive biomarkers and thus fingerprints.

New methods for detecting small signals

1.2 Goals, Objectives and Tasks

As pointed out in the previous section the main goal of this thesis is to find characteristic signals (biomarkers) of a disease in mass spectra of human blood samples. If such a signal is present in a spectrum this could mean that the individual this sample stems from suffers from this disease. Special focus is put on the highly increased sensitivity of detecting the signals in very large amounts of data. Two properties that current algorithms cannot deliver.

This thesis has three main parts that are briefly described below. The first part introduces new methods for the reliable detection of proteomics fingerprints from noisy mass spectra. The second part deals with the application of the newly developed pipeline in biology and in medical studies and shows some examples. In the third part we will describe a new distributed computing framework that allows us to analyze very large amounts of data without the need to implement complicated computer clusters or supercomputers.

Today's mass spectrometry (MS) based protein fingerprinting techniques rely on the analysis of spectra from complex biological protein mixtures (e.g. serum) obtained from high-throughput platforms in clinical settings. The general workflow to extract fingerprints from raw data of two patient groups is:

Fingerprint extraction workflow

1. Detect contained signals (and filter out noise)
2. Evaluate the signals
3. Identify biomarkers (that is statistically significant differences between the groups)
4. Build fingerprints and train classifiers using these fingerprints
5. Test performance (that is classification power) of the resulting classifiers in independent clinical studies

A follow-up study then often tries to determine the underlying molecules to link the fingerprints to e.g. metabolic pathways.

Part I: Detecting Fingerprints

There are many (still unsolved) problems associated with each of these pipeline steps. For example, detecting even smallest but relevant signals in the raw data - which is a complex mix of the real biological signals and (random and systematic) noise introduced by the high throughput MS machines. In the first part of this thesis we propose a solution to this problem: new statistic driven approach that allows to analyze noise and to identify signals below the commonly used signal-to-noise threshold[2] (chapter 3).

Additionall signals identified can be used in subsequent steps to build better patterns for proteomic fingerprinting analysis. We believe that this will foster identification of new biomarkers having not been detectable by most algorithms currently available.

Other very important issues are also addressed, such as preprocessing the raw signals (e.g. to reduce systematic noise, see section 3.3), reliable mapping of detected signals across different spectra to allow comparison (section 3.6), building robust and compact fingerprints (section 3.8) and finally using these fingerprints to classify unknown spectra (section 3.8.5).

Part II: Medical Application

The algorithms and methods developed in this thesis can be combined to an analysis pipeline for automated fingerprint detection from and analysis of mass spectrometry data. This pipeline has been set up and equipped with a web-frontend to allow access for remote scientists (for example in hospitals).

To prove the platform's practical relevance it has been utilized in several clinical studies. We could successfully detect fingerprints for different cancer types (bladder, kidney, testicle, pancreas, colon and thyroid). Two of these studies are presented in chapter 4.

Experiments have shown, that the fingerprints found by our algorithms are missed by commercially available systems that are less sensitive than our approach.

[2]The *thresholding* method only regards signals if their height is above a certain value determined by a noise-estimation step. A common setting for the minimum signal height is three times the estimated noise level.

Part III: Coping with Mass-Data

The analysis of the typically very large amounts of data (up to several Terrabytes per clinical study) is a very computationally intensive task. In chapter 5 we introduce a new framework that allows desktop computers, machines from computer clusters or other special hardware (such as Sony's Playstation 3) to contribute their computing resources in a GRID-like network without the need of installing complex software.

This framework is then exemplarily used by our newly developed algorithms for the spectra data analysis.

Chapter 2

Preliminaries

2.1 Topic Overview

This section briefly introduces the three main topics we deal with in this thesis: Mass Spectrometry (MS) and its applications, proteomics and GRID computing. The basic connection is as follows: Data is acquired through MS-based technology, analyzed in a proteomics context and computed using GRID techniques.

2.1.1 Mass Spectrometry

This section briefly introduces the general mass spectrometry (MS) technology. As we will see below, MS comes in different flavors and each sub-type used introduces different problems and affects some characteristics of the resulting data. The data used and analyzed in this thesis was produced by a MALDI-TOF MS machine and we therefore will focus on this sub-type. In the introduction of Chapter 3 we will give more details of the MALDI-TOF MS process and show the problems and difficulties connected to the MS technology in general and MALDI-TOF MS in particular.

Why MS ?

Rapid growth of projects and studies in the areas of biological, clinical, pharmaceutical, environmental and material sciences have led to dramatically increased demands for chemical and structural information of molecules, especially from complex systems. Mass spectrometry (MS) has become one of the most successful and popular techniques for the analysis of a broad range of analytes. The underlying concept is to ionize the analytes of interest followed by separation according to their mass-to-charge ratio (m/z). Adding an optional separation step preceding the MS step, such as gas chromatography (GC), high performance liquid chromatography (HPLC) or capillary electrophoresis (CE), MS becomes a very powerful tool to even detect compounds from very complicated systems.

Summarized, when using a suited separation technique combined with a specific ionization source and a high resolution mass analyzer, mass spectrometry provides accurate molecular mass data which allows the determination of elemental composition and chemical structure of molecules.

How MALDI-TOF MS Works

The main steps in MALDI-TOF mass spectrometry[1] are:

Sample Pre-fractionation: To reduce the sample complexity, that is the number of different proteins in a sample, it is usually pre-fractionated. Widely used approaches are for example magnetic beads or liquid chromatography (LC).

Magnetic separations of proteins and peptides have been used widely for isolation, separation and purification (Safarik and Safarikova, 2004). Usually, an appropriate affinity ligand is directly coupled to magnetic particles thus exhibiting the affinity towards target compound(s). Then these particles are added to the sample and target compounds bind to them. Subsequently, magnetic particles with isolated target compound(s) are magnetically separated and a series of washing steps is performed to remove the majority of contaminating compounds and particles. Then the target compounds are usually eluted.

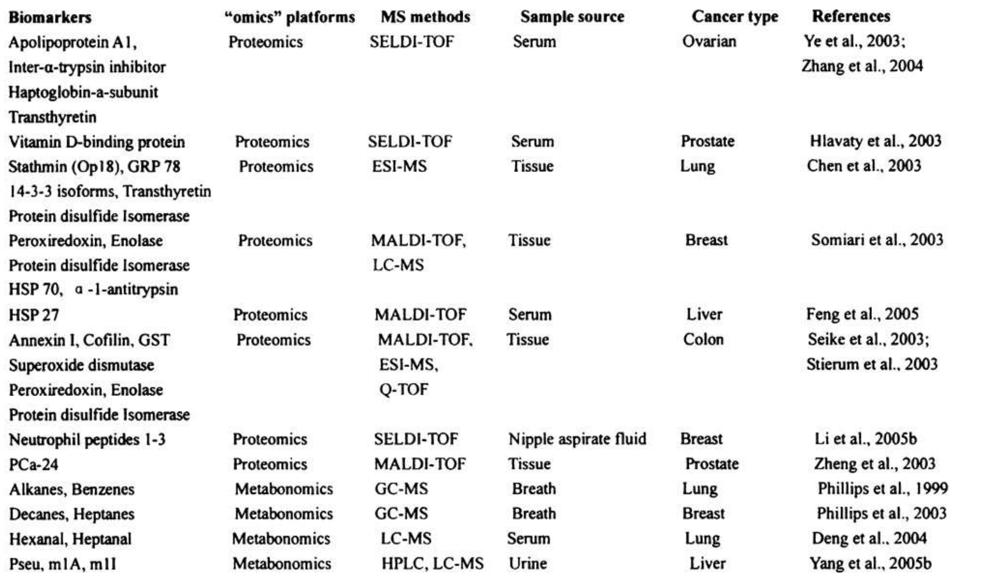
Liquid chromatography (LC) is another fractionation method using a variety of chemical interactions between the molecules in a sample (the analyte) and a chromatography column. Basically, a mixture of liquid (e.g. water) and the actual sample containing the molecules (the mobile phase) is pumped through a column of stationary phase (usually a tube containing small particles having a particular surface similar to the magnetic bead case) at high pressure. The sample/liquid mixture is retarded by interactions with the particles in the column (stationary phase) as it traverses the column. The time a particular peptide/protein needs to travel through the column (elutes) is called *retention time*. Each molecule has an individual retention time which can be calculated quite accurately, thus allowing to draw further conclusions. For an example see section 3.5.

Sample Preparation: In MALDI MS, samples are usually prepared by mixing with an excess amount (about 10^4 fold) of matrix molecules that absorb laser energies. Then this mixture is deposited on a surface and dried. As the solvent evaporates the sample and the matrix co-crystallize. The purpose of the matrix molecules is to isolate the analyte molecules from each other and absorb the intense laser radiation.

Ionizing: The prepared matrix/sample mixture is shot at with a laser beam that vaporizes and propels the molecules in the mixture into the gas phase and subsequently ionizes the neutral analytes in the plume of the excited-state matrix immediately above the sample target. This soft ionization allows for little or no fragmentation of the molecules and usually produces only singly charged ions.

Measuring: The ions are then accelerated within an electric field and travel through vacuum until they hit a detector. The time needed is measured. This MS type is called time-of-flight mass spectrometer and can deal with molecules of almost any mass. MALDI MS has successfully detected large molecules up to 1.5 million Daltons (Da).

[1] A more technical introduction is given in section 3.2.



Biomarkers	"omics" platforms	MS methods	Sample source	Cancer type	References
Apolipoprotein A1,	Proteomics	SELDI-TOF	Serum	Ovarian	Ye et al., 2003;
Inter-α-trypsin inhibitor					Zhang et al., 2004
Haptoglobin-a-subunit					
Transthyretin					
Vitamin D-binding protein	Proteomics	SELDI-TOF	Serum	Prostate	Hlavaty et al., 2003
Stathmin (Op18), GRP 78	Proteomics	ESI-MS	Tissue	Lung	Chen et al., 2003
14-3-3 isoforms, Transthyretin					
Protein disulfide Isomerase					
Peroxiredoxin, Enolase	Proteomics	MALDI-TOF,	Tissue	Breast	Somiari et al., 2003
Protein disulfide Isomerase		LC-MS			
HSP 70, α -1-antitrypsin					
HSP 27	Proteomics	MALDI-TOF	Serum	Liver	Feng et al., 2005
Annexin I, Cofilin, GST	Proteomics	MALDI-TOF,	Tissue	Colon	Seike et al., 2003;
Superoxide dismutase		ESI-MS,			Stierum et al., 2003
Peroxiredoxin, Enolase		Q-TOF			
Protein disulfide Isomerase					
Neutrophil peptides 1-3	Proteomics	SELDI-TOF	Nipple aspirate fluid	Breast	Li et al., 2005b
PCa-24	Proteomics	MALDI-TOF	Tissue	Prostate	Zheng et al., 2003
Alkanes, Benzenes	Metabonomics	GC-MS	Breath	Lung	Phillips et al., 1999
Decanes, Heptanes	Metabonomics	GC-MS	Breath	Breast	Phillips et al., 2003
Hexanal, Heptanal	Metabonomics	LC-MS	Serum	Lung	Deng et al., 2004
Pseu, m1A, m1I	Metabonomics	HPLC, LC-MS	Urine	Liver	Yang et al., 2005b

Figure 2.1.1: Potential Cancer Biomarkers Identified by Mass Spectrometry-Based Omics Technologies, from (Zhang et al., 2007)

2.1.2 MS-based Applications

(Clinical) Diagnostics / Biomarker Detection

The central hypothesis in mass spectrometry-based proteomics technology for diagnosis is that proteins or fragments thereof (peptides) that are characteristic for a particular disease can diffuse into the circulation of e.g. the blood stream. In the case of early diagnosis of cancer, fragments produced by cancer cells or their microenvironment can diffuse into the circulation during tumor development and progression. Their relative concentrations are then measured by mass spectrometry instruments, analyzed by bioinformatics tools and can then be used for diagnosis (see e.g. (Hoffman and Diamandis, 2004; Diamandis, 2004)).

In the past decade, much effort has been put into mass spectrometry-based biomarker discovery and the early diagnosis of diseases, such as cancer (including ovarian, prostate, breast, bladder, renal, lung, pancreas and others). During this time many successful studies have produced potential biomarkers, some of them are listed in Table 2.1.1. In these studies many different *-omics* technologies besides proteomics have been used, such as metabolomics, peptidomics, glycomics, phosphoproteomics or lipidomics with a diversity of components (amino acids, peptides, proteins, metabolites and so forth). Basically, almost any tissue samples or body fluids can be used in these analyses including blood, urine, sputum, saliva, nipple aspirate fluid, tear fluid or cerebrospinal fluid. Although different types of samples need different fractionation methods many of the above mentioned *-omics* technologies can be performed on the same biological samples.

Currently available results are usually presented as a set of discriminating peaks occurring at the same m/z values but having different height (intensities) in normal and cancerous samples. The peptides / proteins underlying these peaks remain unknown but this does not greatly limit its utility for medical diagnostics or classification, because diagnosing a disease is a problem of prediction rather than of etiology (Boguski and McIntosh, 2003).

Further analyses of these discriminating peaks can lead to (at least partial) identification of the underlying peptides by tandem mass spectrometry (MS/MS) and might allow new insights into the actual function of these proteins in the organism (see section 6.2.3 for an example).

Drug Development & Pharmacokinetics

Mass spectrometry based techniques are frequently used in many drug development stages such as “Peptide Mapping”, “Bioaffinity Screening”, “In Vivo Drug Screening”, “Metabolic Stability Screening”, “Degradant Identification” or “Quantitative Bioanalysis” (Lee and Kerns, 1999).

Pharmacokinetics as a branch of pharmacology can be used to analyze the metabolism of substances (e.g. newly developed drugs) brought into a living organism. MS based pharmacokinetical studies can help determining how quick a drug will be cleared from the Hepatic Blood flow and organs of the body. Advantages of MS based technologies over commonly used methods such as UV based detection techniques are its high sensitivity and specificity and a very short analysis time (see e.g. (Covey et al., 1986; Hsieh and Korfmacher, 2006)).

Peptide Quantification

Peptide quantification is a method to compare the amounts of a given protein among two or more samples. (For an introduction see e.g. (America et al., 2006; Wang et al., 2006; Wang, Zhou, Lin, Roy, Shaler, Hill, Norton, Kumar, Anderle and Becker, 2003).) There are two main strategies for quantification:

Label-based approaches (e.g. ICAT, MeCAT, SILAC) use chemical labels to mark individual peptides as coming from one or the other sample and measure the difference in one MS run. For example additional neutrons are added to one of the samples by introducing stable isotopes in one or more of the atoms comprising the peptide under scrutiny (e.g., ^2H replacing ^1H). The peak intensities of the respective peptides can then be compared to obtain a relative quantification of the peptide in sample 1 vs. sample 2. See (Ong and Mann, 2005) for a good overview of different labeling strategies.

Label-free approaches depend on comparison of two (or more) individual MS runs of the same peptide (in different samples) and comparing the peak intensities. Obviously, two main critical factors are the mass accuracy of the mass analyzer used and the ionization quality of the sample, since the quantification depends on the identification of peaks belonging to the peptide. The main advantage of this approach is the low cost, since peptides do not have to be biochemically modified.

2.1.3 Proteomics

The neologism “omics” refers to some (usually but not only) biological field of study such as *genomics* or *proteomics*. The actual object of this study is then built with the related neologism “omes”, such as the *genome* or *proteome*. Most of the first scientists to use the “-ome” suffix were Bioinformaticians and molecular biologists who wanted to refer to some sort of wholeness or completeness and thought it has some roots in the Greek language - a fact that was never confirmed in Greek literature.

However, *proteomics* (in analogy with *genomics*) has become a well established term for (large-scale) studies of proteins, in particular studies of their structures and functions. Proteins are the main components of the physiological pathways of cells. Similar to the *genome* as the set of genes of an organism, the *proteome* is the set of proteins produced by it during its life.

For many scientists, proteomics is considered the next step in the study of biological systems, logically succeeding genomics. Although the genome is rather static the proteome is very dynamic: it differs between cells (even of the same type) and constantly changes (or adapts) in response to biochemical interactions with the genome and changes in the environment. The conclusion that follows from these findings is that one organism will have totally different proteomes at different stages of its life.

Another major difficulty in proteomics is the vast amount of existing proteins and their derivatives (estimated over 500.000) that are due to mechanisms such as alternative splicing or posttranslational protein modifications (e.g. glycosylation or phosphorylation).

2.1.4 Grid Computing

Grid computing can be defined as an expandable, scalable set of resources applied to solve a single or a set of problems. These are usually scientific or technical problems that require a large number of single, non directly dependent calculations. There are a lot of application domains where grids can be used in, such as design of integrated circuits, drug discovery, molecular modeling, financial simulations, analysis of biological data, or any number of other computational intensive calculations.

A grid is an architecture that enables dynamic allocation of resources to varying workloads in accordance with computational needs. Users do not care where the compute cycles or data reside - it is delivered to them quickly, efficiently and seamlessly.

Homogeneous Grids (that are similar to the well-known computer *clusters*) are usually built with (almost) identical low cost modular components, so one can start and increase the number of modules as the need grows. In this case *identical* means the type of architecture (e.g Intel's Pentium 4) and operating system (OS, e.g. Linux Debian).

The roots of grid computing came from the university and scientific communities in the 1980s and early 1990s. It has grown from a viable option to a necessity for companies in High Performance Technical Computing.

Heterogeneous Grids

Opposed to a homogeneous grid described above, where all modules (machines) are of the same type (e.g. Intel's Pentium 4 machines running Debian Linux 3.4) a heterogeneous grid consists of different combinations of machine architectures (Intel, Sun, Sparc, Macintosh, IBM Cell / Playstation 3 - see Section 6.2.4, ...) and operating systems (Windows, Linux, Unix, Mac OS, ...). This is typically the case in a mixed environment like a university network. One therefore needs a special type of software to be able to

- Use the same algorithms on these different architectures without the need of re-implementation

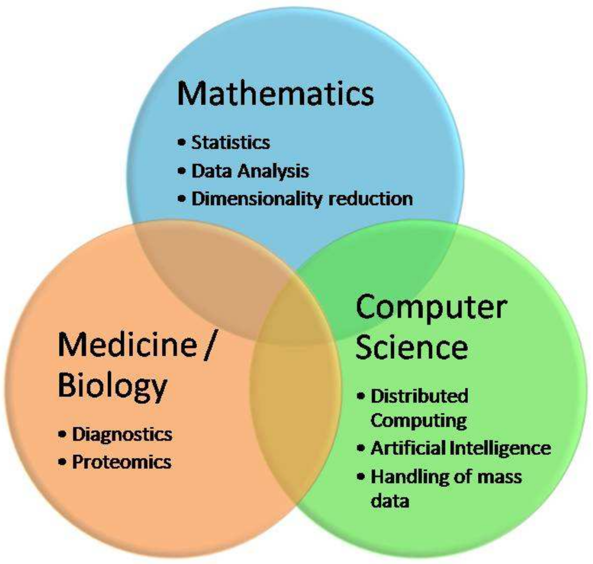


Figure 2.1.2: The different fields and some sub-areas we are using in this thesis.

- Provide a way to efficiently exchange (or access) data on these different platforms and file systems
- Set up communication schemas to distribute the single calculations to the particular machines and monitor the execution.

In Chapter 5 we describe the approach we have developed to solve these problems and introduce our workflow-based worker system (section 5.2.3).

2.1.5 Combination of These Topics

The previous sections briefly introduced the different topics this thesis will deal with: mass spectrometry-based proteomics, clinical diagnostics (requiring data analysis) and grid computing. Figure 2.1.2 schematically shows how these topics are inter-connected. The common thread that will show up throughout this thesis is of course the analysis of mass spectrometry data to allow clinically relevant diagnosis. The next chapters - which are also suited to be read separately - will first explain how mass spectrometry data is acquired and analyzed (chapter 3) and how these techniques can be used in medicine / clinical applications (chapter 4). Then we explain the bottleneck of the data analysis, namely the large amounts of data that need to be analyzed to get significant results and show how to solve this issue by performing the analyses on many computers in parallel (chapter 5). Finally, we present a prototype and give some case studies to demonstrate the possibilities of these new algorithms and technologies (chapter 6).

2.2 An Example

This section will give a small example to illustrate the main steps of fingerprint detection, introduced in the previous section. To avoid scientific jargon as far as possible we have chosen to do this on the basis of piles of Lego bricks

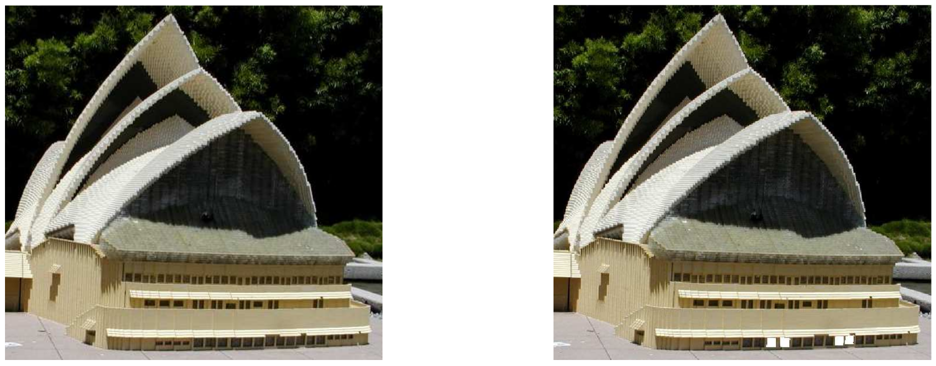

(a) Opera A (b) Opera B

Figure 2.2.3: The two types of operas to be built. Note the little difference at the doors of the opera: in the right (type B) there are four white doors.

(which might seem a little bit artificial but serves the purpose). The connection between Lego piles and spectra is quite simple: both contain several thousand signals - single Lego bricks in a pile or peaks in a spectrum. Again, the challenge is to find signals that can be used to differentiate between two groups (piles "A" vs. piles "B" or spectra group "diseased" vs. spectra group "healthy"). However, spectra of two different groups might differ in just one or two changes in the signals. In other words, the images of two spectra are extremely similar.

Suppose at LEGOLAND they want to build a new Lego opera city consisting of two different kinds of operas (opera A and opera B, see Figure 2.2.3). For some reasons, they want the two types to look almost identical and therefore the required Lego bricks are almost the same for each type. The LEGOLAND officials already have a big storage building where a machine has produced all the needed Lego bricks: one big pile for each opera to build (as shown in Figure 2.2.4). All the piles look virtually identical since there existed a production schema specifying the order in which the bricks are to be produced and where to place them. The machine exactly followed this plan, so the only way in which two piles can differ is the type of bricks (between an opera-A pile and an opera-B pile) or if a brick slid down a pile's slope and therefore its desired position got altered.

Unfortunately, this production plan is lost, hence, we have no idea where the differences between the two kinds of piles are. All we know is that the differences are extremely small, as shown in Figure 2.2.5: here we zoomed into the same region of an opera-A and an opera-B pile. Luckily, when the machine produced the piles, someone did label some of them with their respective labels, "opera-A" or "opera-B".
The tasks are now:

1. to identify which Lego bricks distinguish the two groups of piles and

2. to tag each of the remaining unlabeled piles with its appropriate description (A or B).

To enable us doing the analyses the LEGOLAND management has sent us pictures showing every pile from top (see Figure 2.2.6 for an example). Obviously, the camera used was not very good so the pictures taken are quite

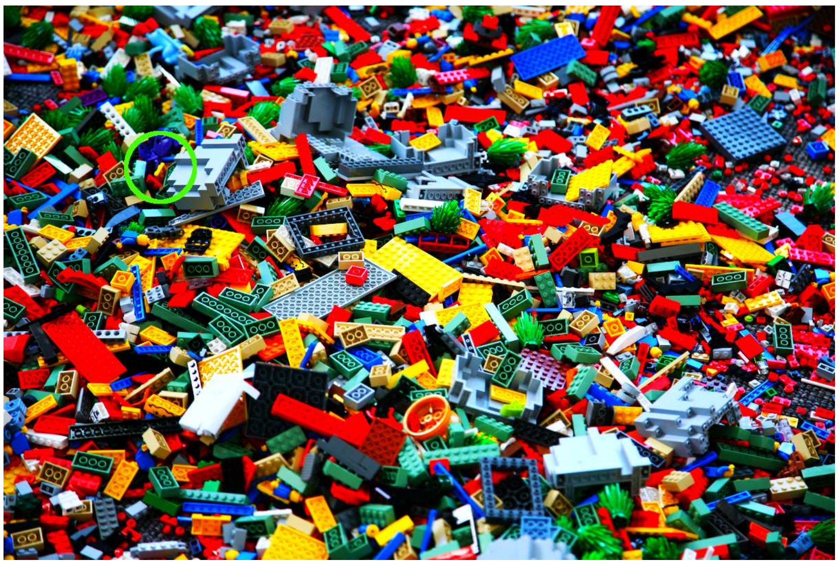
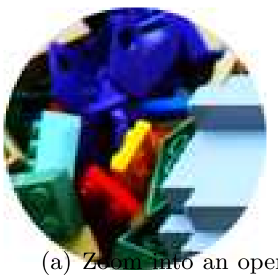
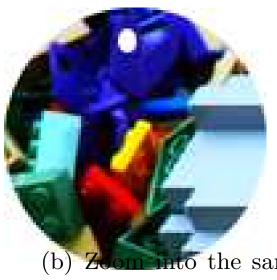


Figure 2.2.4: An exemplary pile of Lego bricks. These could be the Lego bricks needed to build a type A or type B opera.



(a) Zoom into an opera-A pile - the encircled area in picture 2.2.4.



(b) Zoom into the same region as in the left picture of an opera-B pile.

Figure 2.2.5: Zoomed into the same region of Lego piles for the two different operas.

Figure 2.2.6: Two pictures of two distinct piles of Lego bricks taken by the LEGOLAND team. They differ in one Lego brick. (Hint: it is the brick shown in Figure 2.2.5.)

blurred. This extremely complicates the reliable detection of details such as the borders of the single bricks or small differences. The two pictures in Figure 2.2.6 have the same differences as in Figure 2.2.5 - can you still find them ?

Recipe

Now, having an idea about the data we are working with (in this example) we will give a small recipe to fulfill the task stated in the previous section. We will later use the ideas sketched here to describe the actual algorithms developed in this thesis. The steps are:

1. Find the bricks
2. Group the bricks
3. Analyze the groups
4. Check the feature quality
5. Compile a fingerprint

Step 1: Find the bricks

In each picture detect the borders of the individual Lego bricks (see Figure 2.2.7). For each brick found, write down its position and color in a list. Since every picture can be processed independently from each other we can distribute the single pictures to many different workers and collect the resulting lists afterwards.

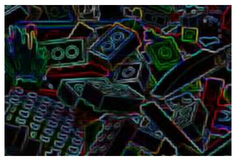


Figure 2.2.7: Shown are the borders found by a border detection algorithm applied to a picture of Lego bricks.

Step 2: Group the bricks

Use the lists of the previous step to identify and group together the same brick in each picture. We assume that the same brick will appear at almost the same position in each pile.

For example, bricks of group 1 have the position shown in Figure 2.2.8: notice that although the two bricks of group 1 differ in color (the little white dot alters the bricks overall color) they still belong to the same group. So, a group is essentially a list of bricks having the same (or similar) positions identified by a specific number.

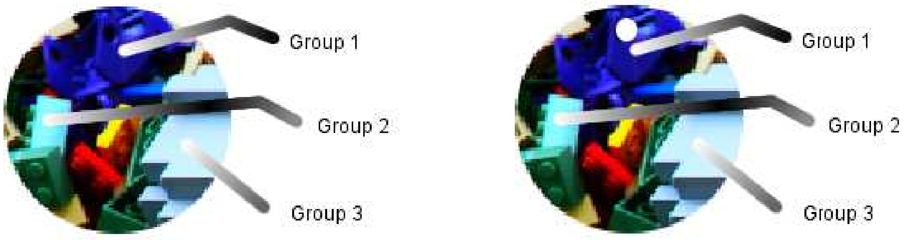
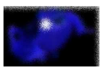
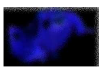


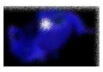
Figure 2.2.8: Bricks of the two kinds of operas. Some of them are labeled with the number of the group they belong to. Notice the difference in the two bricks of group 1 and recall that the only condition for a group membership is the position while other properties such as color may be different.

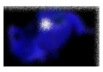
Step 3: Analyze the groups

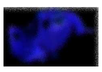
For each group from the previous step check if all its bricks roughly look the same or if we can identify differences between bricks from an opera-A pile versus an opera-B pile. For example, within the same group bricks from an opera-A pile might be darker than bricks from an opera-B pile which would then be two sub-groups with different average color. This is illustrated in Figure 2.2.9: we are looking at seven random members of group 1 (see Figure 2.2.8) and there are obviously two sub-groups: the bricks from opera-B type piles have a white spot so the average color of a brick in this opera-B sub-group is around 230. Bricks from an opera-A type pile do not have this white spot which decreases the average brightness value to about 200.

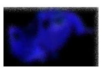

(a) B, color: 231


(b) A, color: 205


(c) B, color: 134


(d) B, color: 232


(e) A, color: 204


(f) A, color: 206

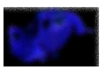

(g) A, color: 201

Figure 2.2.9: Seven lego bricks of group 1 in Figure 2.2.8. Obviously, there exist two sub-groups.

From now on we will call a group that has two different sub-groups a *feature*. So here a feature has three properties: the group number and the average color values of its two sub-groups (for group 1: 200 for the opera-A type and 230 for the opera-B type).

Now, the usage of a feature to determine an unknown pile's type is quite simple: From our feature we know the average position (since it is still a group) and the color a brick would have if it belongs to an opera-A or opera-B type pile. Consequently, we just have to take the picture of the unknown pile, identify the brick at this features position and determine its color and compare it to the colors for an opera-A or an opera-B type. The closer it is to one of them, the more likely it is that kind of opera.

Step 4: Check the feature quality

Use each feature found in the previous step to guess the label of a pile we already know. If the answer is correct, we count this as a success. We now test each feature with every (already labeled) picture we have and count the successes. We call the percentage of successes the quality (e.g. 30 correct labels out of 50 pictures yields a quality of 60%).

Step 5: Compile a fingerprint

Out of the list of features from the previous step take all features having a quality value above 95%. We call this resulting list of features a fingerprint. To determine the type of a pile using such a fingerprint, we perform the decision as described in step 4 for each feature of this fingerprint one after another. Each decision is treated as a vote for opera-A type or opera-B type. The type with the most votes wins and is taken as the final result.

Remarks

Of course this example and the recipe are quite simple and sketchy. We omitted many details, background information and occurring problems - for example, how do we handle pictures with missing areas ? And, admittedly, it is not very common that one needs to tell apart two piles of Lego bricks on the basis of fuzzy pictures.

However, in the next chapters we will see how similar these problems, questions and eventually concepts and solutions are in the actual field of this thesis: the analysis of groups of mass spectra images resulting from human blood samples. We will show how to use, extend and improve these concepts.

2.2.1 Term Comparison

This table shows a comparison of terms used in the Lego example and the according terms used in later chapters.

Term in Lego example	Term used later
Pile picture	Mass spectrum
Storage building	Hard disk drive
Lego brick producing machine	Mass spectrometer
Camera	Time-of-flight measurement
Opera type (A and B)	Patient group (e.g. *healthy* or *diseased*)
Brick	Peak
Brick position	Peak center
Brick color	Peak area and shape
Brick border	Peak shape
Feature	Feature
Fingerprint	Fingerprint

Chapter 3

Mathematical Modeling and Algorithms

Contents

3.1 Introduction

In the introductory Lego example (section 2.2) we presented the core idea of this thesis: given two different groups of Lego piles, find bricks that correspond in piles within the same group but differ between the two groups. We called those bricks *features* and used the best features to compile a *fingerprint*. A fingerprint could then be used to identify the type of an unknown pile. Although being quite simple we already identified some fundamental challenges, such as the identification of a bricks borders given fuzzy data.

When transferring the ideas of the Lego example to the analysis of mass spectra the main idea stays the same: find distinguishing properties of two groups. As we will see in chapter 4 these properties can be used to build classification algorithms able to distinguish healthy from diseased people. This chapter will describe the pipeline necessary to achieve this which is shown exemplarily in Figure 3.1.1. As the colors indicate, there a four basic components:

Preprocessing: Prepares the spectrum for subsequent stages. These preprocessing steps mainly removes noise and systematic errors (section 3.3).

Peak Seeding: Find signals (peaks) in the preprocessed spectrum (sections 3.4, 3.5). This is similar to the *border detection* step in the Lego example

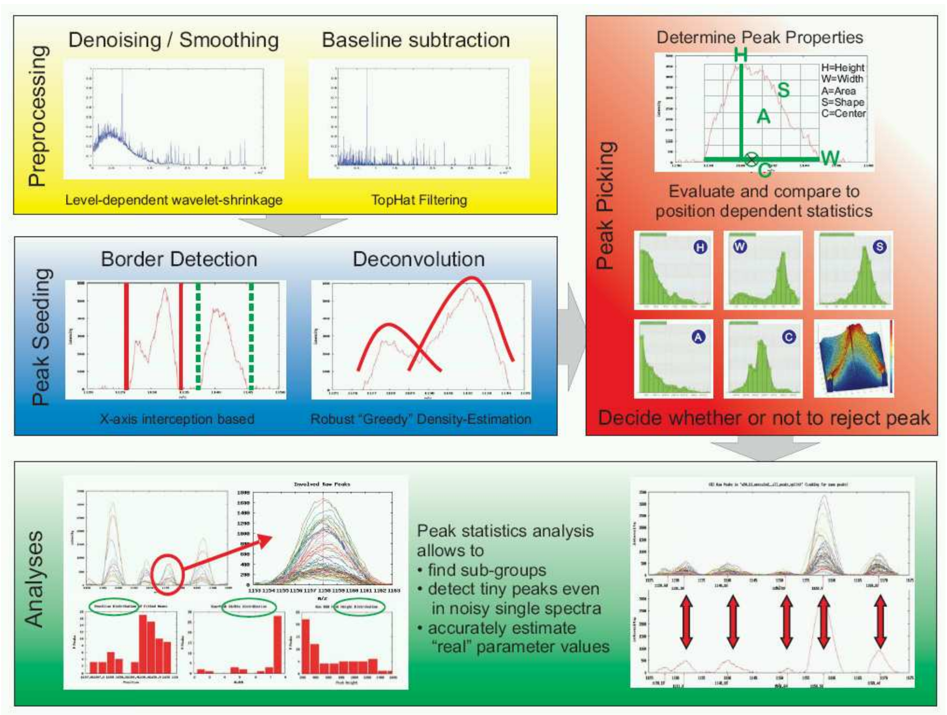

Preprocessing
Denoising / Smoothing
Baseline subtraction
Level-dependent wavelet-shrinkage
TopHat Filtering
Peak Seeding
Border Detection
Deconvolution
X-axis interception based
Robust "Greedy" Density-Estimation
Peak Picking
Determine Peak Properties
H=Height
W=Width
A=Area
S=Shape
C=Center
Evaluate and compare to position dependent statistics
Decide whether or not to reject peak
Analyses
Involved Raw Peaks
Peak statistics analysis allows to
• find sub-groups
• detect tiny peaks even in noisy single spectra
• accurately estimate "real" parameter values

Figure 3.1.1: Overview of the pipeline. The colored boxes show the distinct modules: "(Signal) Preprocessing", "Peak Seeding", "Peak Picking" and "Analysis". The gray arrows indicate the data flow.

(see section 2.2): at this stage borders of the peaks are detected and convoluted peaks separated.

Peak Picking: Classify signals found as noise or relevant peaks (sections 3.4, 3.6). This corresponds to the grouping of bricks in the Lego example.

Analyses: Determine significance of relevant peaks to distinguish spectrum groups, such as "healthy" vs. "diseased" (section 3.7). Select the best peaks to distinguish between two groups (section 3.8).

Now, one of the first questions jumping to mind when looking at a technology, some hundred years old usually is: "Isn't it already fully understood and haven't algorithms been developed already that can perfectly analyze these data ?" The simple answer is "No". It is still unclear what exactly happens inside the machine (with respect to physical and biochemical phenomena) and many approaches have been published that are continuously improving, but still far from being perfect. To give you a better understanding, the next section will introduce the main principles and resulting problems we have to deal with. In the remaining of the chapter we will introduce our approaches to cope with these issues.

3.1.1 Comparison to Other Concepts

The division into the components (pipeline steps) described above is somewhat different to what is usually described in the literature. This is because most algorithms are using one spectrum at a time for analysis in contrast to o ur approach where we use many spectra simultaneously to refine the quality of our findings. For example, usually a *Peak Picking* algorithm is designed to extract the precise mass/charge (m/z) ratio of the contained peptides and comprises the steps we named *Peak Seeding* and *Peak Picking*.

As will will see in later chapters this approach seems reasonable to us since the majority of data used in this work[1] is acquired from highly complex protein mixtures. Most other algorithms are designed to analyze data stemming from less complex samples. Thus, also the concept of a *Peak* is slightly different: in this work we use the peaks found during an analysis to de-convolute larger peak groups to obtain the original peak patterns representing single peptides. In data from less complex samples the de-convolution can usually be ommited and a peak (or a small group of peaks called an *isotopica pattern* see section 3.4) can typically be identified with a particular peptide.

3.2 Introduction to MALDI TOF MS

Introductory remark: A complete understanding of the internal physical or (bio-)chemical processes of MALDI does not yet exist, which not only affects further optimization of MALDI but also renders a full understanding of the resulting spectra impossible. Poor shot-to-shot and sample-to sample reproducibility resulting from the crystalline matrix (see below) is another issue that must be dealt with when interpreting the results. Finally, due to matrix fragments the MALDI process produces a large amount of ions below m/z 600 (background noise), which makes it impossible to analyze molecules below this threshold.

History of MS

In the mid-nineteenth century, the physicist Julius Plücker investigated light emitted in gas-filled tubes (*discharge tubes*) arising from ionizing the gas by applying voltage through electrodes at both ends of the tubes.

Later, in 1886, another physicist Eugen Goldstein discovered that discharge tubes with a perforated cathode emit a glow at the cathode end. Goldstein concluded that there exists a ray of positive ions passing through the channels in the cathode, which he called *Kanalstrahlen* (canal rays).

In 1899 the Nobel Prize laureate Wilhelm Wien (again, physicist) found that strong electric or magnetic fields deflect these canal rays and constructed a device with parallel electric and magnetic fields. This device could separate positive rays according to their charge-to-mass ratio (e/m) and was further improved consecutively by J.J. Thomson, A.J. Dempster (1918) and F.W. Aston (1919) to create the first mass spectrograph.

The modern version of these MS machines (we are mainly using in this thesis) was introduced in 1987 by Franz Hillenkamp and Michael Karas: matrix-assisted laser desorption/ionization mass spectrometry (MALDI-MS).

In 2002, the Nobel Prize in Chemistry was awarded to John B. Fenn for the development of electrospray ionization (ESI) and Koichi Tanaka for the development of soft laser desorption (SLD) in 1987.

General Principal

As briefly discussed in section 2.1.1 a modern MALDI-TOF mass spectrometer contains at least (see Figure 3.2.2)

[1]The data is described in detail in section 4.1

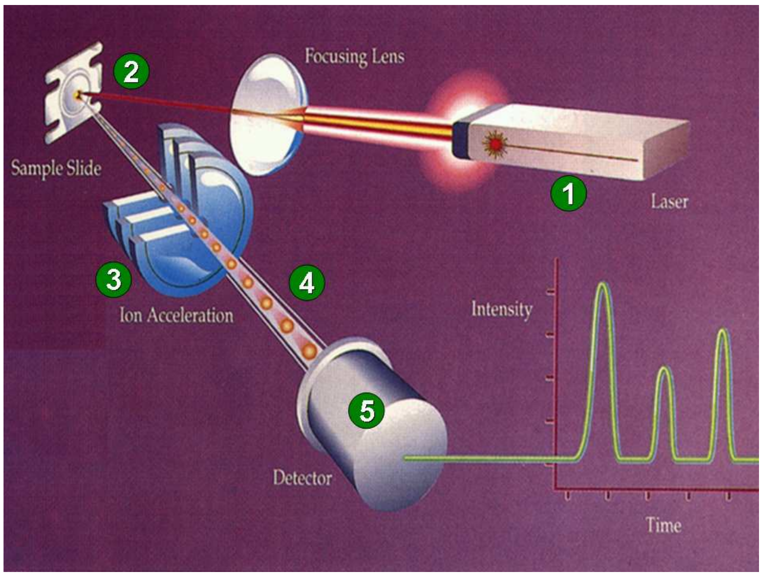


Figure 3.2.2: Basics of a modern MALDI-TOF mass spectrometer. (1): Laser, (2) Sample Slide, (3) Acceleration Chamber, (4) Drift region, (5) Detector. (Picture modified from (Finnigan, 2007).)

- a laser (1) and the matrix/sample mixture (2), serving as the ion source,
- a time-of-flight (TOF) analyzer to separate the ions based on their mass/charge ratio (m/z) (3,4), and
- a sensor for detecting the ions (5).

(2)-(4) and sometime (1) usually occur under vacuum conditions, since collision with residual gas molecules would hinder the ion separation.

q: Charge, $q = e \cdot z$

E: Strength of electric field

The fundamental idea is that the ions (of charge q) are generated by a laser beam ((1) & (2) in Figure 3.2.2) and then accelerated through an electric field of constant energy E ((3) in Figure 3.2.2) , of hundreds to thousand of volts. This allows the description of the ion behavior with newtonian equations (Guilhaus, 1995).

V: Potential

F: Force acting on ions during acceleration

e: Unit charge (charge of a proton)

z: Number of unit charges of ionized molecule

q: Charge

a: Acceleration

Therefore, the ions travelling in this field (with potential V) are accelerated with force F:

$$F = E \cdot e \cdot z = E \cdot q \qquad \text{and} \qquad F = m \cdot a$$

so acceleration (a) equals

$$a = \frac{E \cdot q}{m}$$

v: Velocity

t: Time

s_a: Distance of effective acceleration

v_0: Initial velocity

Acceleration is also the change of velocity (v) over time (t), dv/dt. So in the acceleration region (over distance s_a) with a given initial velocity (v_0):

$$v - v_0 = \int \frac{E \cdot q}{m} \mathrm{d}t$$

and

$$v = v_0 + \frac{E \cdot q}{m} \cdot t$$

t_a: Time ion travels through acceleration field

The time to traverse the acceleration region (t_a) is given by:

$$\begin{aligned} t_a &= \frac{v - v_0}{a} \\ &= m \cdot \frac{v - v_0}{E \cdot q} \\ &= m \cdot \frac{v - v_0}{E \cdot e \cdot z} \\ &= \frac{m}{z} \cdot \frac{v - v_0}{E \cdot e} \end{aligned} \tag{3.2.1}$$

The travelled distance (s) during this time, which is the distance to a zero position, measured from the initial position (s_0) of the ion is calculated by:

s: Distance

s_0: Initial position of ion, relative to zero position

$$\begin{aligned} s - s_0 &= \int v \mathrm{d}t \\ &= v_0 \cdot t + \frac{E \cdot q}{2 \cdot m} \cdot t^2 \end{aligned}$$

After the acceleration through potential V and before the ions eventually hit the detector ((5) in Figure 3.2.2) they travel through the drift region (with length D, (4) in Figure 3.2.2) with kinetic energy U_D and drift velocity v_D and with

D: Length of drift region

U_D: Energy of ion in drift region

v_D: Drift velocity

$$E_{kin} = \frac{1}{2} \cdot m \cdot v^2$$

we get

$$U_D = q \cdot V = q(E \cdot s_a) = \frac{1}{2} \cdot m \cdot (v_D - v_0)^2$$

This allows for the calculation of the drift velocity (v_D):

$$v_D = v_0 + \sqrt{\frac{2 \cdot q \cdot E \cdot s_a}{m}}$$

and therefore we can compute the time the ions travel through the drift region:

$$\begin{aligned} t_D &= \frac{D}{v_D} \\ &= D \cdot (\sqrt{\frac{m}{2 \cdot q \cdot E \cdot s_a}} + \frac{1}{v_0}) \\ &= D \cdot (\sqrt{\frac{m}{2 \cdot q \cdot V}} + \frac{1}{v_0}) \\ &= D \cdot (\sqrt{\frac{m}{z}} \cdot \sqrt{\frac{1}{2 \cdot e \cdot V}} + \frac{1}{v_0}) \end{aligned} \tag{3.2.2}$$

Summarized, the total time-of-flight (TOF) is

TOF: Time-of-flight

$$TOF = t_a + t_D$$

Of course, this is the assumption for a perfect world. In practice there are (at least) two more variables: the time between the start of timing and the acceleration of the ions (t_0) and the detector response time (t_d). Including

t_0: Time of ion formation

t_d: Detector response time

this yields:

$$TOF = t_0 + t_a + t_D + t_d$$

Modern TOF MS machines usually ensure that $v_D \gg v_0$ and t_0 and t_d are as small as possible. Nevertheless, these small contributions to the overall TOF have an impact on the spectra as we will see later. However, following from equations 3.2.1 and 3.2.2 we can state that:

$$TOF \propto \sqrt{\frac{m}{z}} \tag{3.2.3}$$

This exhibits the actual relation between mass and time-of-flight:

$$\frac{m}{z} = a \cdot TOF^2 + b \tag{3.2.4}$$

a being a proportionality constant that can be shown to be

$$a = \frac{2 \cdot s_a \cdot e \cdot U_D}{(2 \cdot s_a + D)^2}$$

and b another constant modeling the influence of t_0 and t_d (and potentially others). Of course, equations 3.2.3 and 3.2.4 only hold if all accelerations are constant during measurement.

The beauty of these constants is that they can be used to calibrate a mass spectrometer by determining their (machine dependent) values from the times of flight of some known m/z values.

Resolution Issues

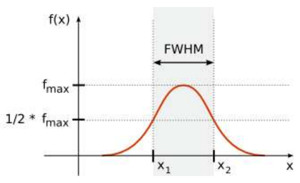


Figure 3.2.3: This shows the definition of peak width: the width of a peak is defined to be the width the peak has at its half maximum (Full Width At Half Maximum, FWHM).

The resolution of a mass spectrometer states the ability to separate ions of similar mass-to-charge ratio. It is defined as $\frac{\frac{m}{z}}{\triangle\frac{m}{z}}$ where $\frac{m}{z}$ is the value of interest and $\triangle\frac{m}{z}$ the width of a peak at this $\frac{m}{z}$ value at half maximum height (full width at half maximum height, FWHM, see Figure 3.2.3). Intuitively, it is clear that the narrower a peak for ions of a particular $\frac{m}{z}$ ratio, the better the resolution of the machine. To show this let us start with Equation 3.2.4:

$$\tfrac{m}{z} = a \cdot t^2$$

Differentiation with respect to time yields:

$$\frac{\mathrm{d}\frac{m}{z}}{\mathrm{d}t} = 2 \cdot a \cdot t$$

For a finite interval this becomes:

$$\triangle\frac{m}{z} = 2 \cdot a \cdot t \cdot \triangle t$$

and hence

$$\triangle\frac{m}{z} \cdot t = 2 \cdot \frac{m}{z} \cdot \triangle t$$

$$\frac{\frac{m}{z}}{\triangle\frac{m}{z}} = \frac{t}{2 \cdot \triangle t}$$

The final equation clearly shows that resolution mostly dependends on the difference in the measured flight times for ions of similar mass or m/z values.

Or, in other words, the more exact the machine can determine the time an ion travels from the moment of its ionization until it hits the detector, the higher the resolution is. Analyzing the equations from the previous section we see that many factors contribute to an ions flight time, including:

1. initial states of the ions (prior to acceleration), such as velocity (v_0), position (s_0) and time of formation (t_0)
2. non-ideal vacuum that introduces particle collisions (affects u_D)
3. non-ideal acceleration or drift regions, which affect u_D and U_D
4. fragmentation of ions, either in acceleration or drift region (affects at least u_D)

The factors that limit the resolution mostly are the distribution of the ions initial states. These result in slightly blurred spectra (see the Lego example).

Other Types of MS

In this thesis we mainly use data from MALDI MS machines since most of the recent publications in clinical setting are using this kind of technology. However, besides MALDI there are other types of MS technology that are briefly described in the following paragraphs.

SELDI Surface-enhanced laser desorption/ionization (SELDI) is another common ionization method in mass spectrometry that is used for the analysis of protein mixtures (Tang et al., 2004). Opposed to MALDI, the protein mixture is spotted on a surface with a chemical. Such a surface (e.g. CM10, a weak-positive ion exchange, or IMAC30, a metal-binding surface) specifically binds some proteins from the sample while the others are removed by washing. After washing - as in MALDI - the matrix is applied to the surface and crystallizes with the sample peptides. Thus, the SELDI surface acts as a separation step. Surfaces can also be built with antibodies or DNA.

SELDI technology was developed by T. W. Hutchens in 1993 (Hutchens and Yip, 1993) and commercialized by Ciphergen Biosystems in 1997 as the ProteinChip system. It is now produced and marketed by Bio-Rad Laboratories.

ESI Electrospray ionization (ESI) (Fenn et al., 1989) is another technique to produce ions and rewarded with the Nobel Prize in Chemistry to J.B. Fenn in 2002. It is especially well suited to ionize *large* macromolecules because it overcomes the propensity of these molecules to fragment during ionization.

In electrospray ionization, a liquid containing the sample is pushed through a very small, charged capillary (Fenn et al., 1990). The sample is dissolved in a large amount of solvent and is already ionized. Because like charges repel, the liquid pushes itself out of the capillary and forms an aerosol consisting of small droplets. As the solvent evaporates, the sample molecules are forced closer together, which causes repellence thus breaking up the droplets. Resulting ions are then accelerated and fly to the detector as in the SELDI or MALDI case. Opposed to the MALDI ionization process ESI produces also multiply-charged ions such as $[M+2H]^{2+}$.

3.3 Preprocessing

3.3.1 Introduction

The raw data $X(t)$ acquired from the Mass Spectrometer (MS) is a mixture of the real signal (S) and multiple components of noise (see Figure 3.3.4): $X(t) = B(t) + I \cdot S(t) + \epsilon(t)$, which are described below. The preprocessing step removes these modifications and recovers the original data as good as possible. These steps are crucial for subsequent algorithms (such as peak detection or further analyses) to deliver reliable results.

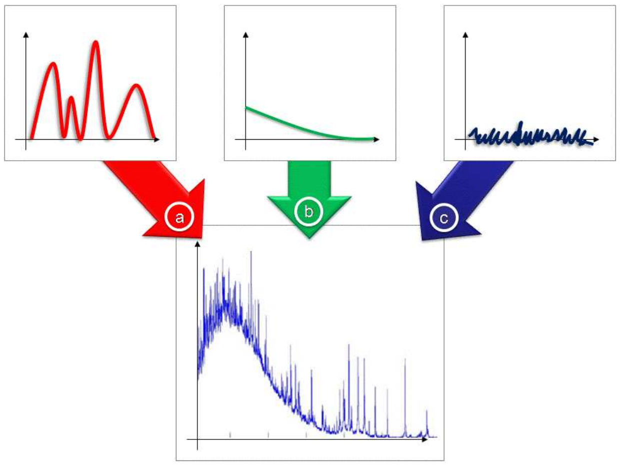


Figure 3.3.4: Components of the raw data acquired from the MS machine: (a) actual signal (S), (b) baseline (B) and (c) random and chemical noise (ϵ).

In the Lego example (see section 2.2) the error source we considered was the fuzziness of the pictures leading to problems in border detection.

3.3.2 Sources of Uncertainty

In this section we describe the sources of the different types of noise and transformations which modify the original data and eventually produce the raw data that is acquired by the MS machine, these are:

Systematic and random noise (B, ϵ) The noise itself consists of a low-frequency baseline (B) and high-frequency chemical and random noise (ϵ). The baseline is an exponential like offset dependent on the m/z value (mass-to-charge; x-value). It is mainly caused by clusters of matrix components and small molecular fragments originating from degradation processes, desorption and collisions in the acceleration phase.

Incomplete (missing) data After the detector of the MS machine is hit by a particle it needs some milliseconds to recover. During this time it cannot record other particles it collides with and is literally blind. However, advances in the detection techniques and new MS technologies like ion-trap seem to decrease this type of error in the future.

Shifts in m/z direction A MS machine needs to be calibrated before it can be used for the acquisition of data. This calibration needs to be done carefully and many factors like temperature, humidity and the peptide mix (external standard) used are important for reliable results. Not only the zero-point of the particular machine is determined, but also machine specific transformations (machine dependent constants, see for example Equation 3.2.4). Once these parameters are determined correctly the measurement errors are corrected automatically during acquisition within the machines hardware.

Shifts in intensity direction (I) A biological sample (e.g. blood) automatically undergoes some changes in its biochemical content, mainly caused by proteases (see section 4.1.1). Furthermore, when mixed with the so-called matrix, a quite inhomogeneous mixture forms, which then becomes the final sample, put into the MS machine. Within this inhomogeneous sample there exists so-called *sweet spots* where the density of proteins is much higher than average. If now the laser beam hits these sweet spots, the intensity of these molecules increases excessively.

3.3.3 Our Approach

In the following, we describe the algorithms we have developed (or modified to fit our needs) to step-by-step recover the original signals available in the sample put into the MS machine. The overall procedure is as follows:

Algorithm 1 Preprocessing

```
Require: Raw Spectrum as $x, y$ value pairs
  Apply wavelet-based de-noising
  Apply tophat-based baseline reduction
  Apply normalization
  return Preprocessed spectrum
```

3.3.4 Denoising

Denoising the raw data X tries to remove whatever noise (ϵ) is present in X while retaining whatever signal S is present. (Note baseline removal baseline is handled separately - see section 3.3.5.) This is not to be confused with *smoothing* which removes high frequencies present in the data and retains low ones opposed to denoising which attempts to remove whatever noise is present. Denoising generally yields better results in subsequent steps of the analysis workflow, since some general assumptions about smoothness can be taken. However, in practice most of the noise in MALDI-TOF spectra is indeed contained in the high-frequency component of a spectrum. This is mainly due to a number of factors, such as electrical inference, random ion motions, statistical fluctuation in the detector gain or chemical impurities (see e.g. (Shin et al., 2007)). There are several (heuristic) approaches for noise reduction in the literature such as moving average filters (Liu, Krishnapuram, Pratapa, Liao, Hartemink and Carin, 2003), Gaussian kernel filters (Wang, Howard, Campa, Patz and Fitzgerald, 2003), or PCA (Statheropoulos et al., 1999). However, most of these (parametric) noise reduction approaches have been established based on empirical insights and the parameters need to be fine-tuned to make the method work properly - this is always case sensitive and time consuming.

Our studies have shown that the approaches mentioned above are not very well suited for reducing noise in MALDI/SELDI-TOF spectra: for example, in some of our test cases they alter the relative peak intensities where subsequent analyses rely on (e.g. to obtain the relative compound concentrations correctly). This has also been found by by others (e.g. (Li et al., 2007)). Our preliminary experiments have shown that non-parametric techniques are more appropriate. Out of these approaches a wavelet shrinkage approach (for a review see (Taswell, 2000); a nice introduction can be found here (Louis et al., 1998)) has performed best on artificial data and data from spiked experiments where we knew the peak positions and intensities. This wavelet shrinkage approach has been reported to be very well suited for denoising of mass spectrometry data (see e.g. (Ojanen et al., 2004; Liu, Sera, Matsubara, Otsuka and Terabe, 2003; Coombes et al., 2005)).

Opposed to other denoising algorithms, such as moving average or low-pass filter (e.g. Savitzky-Golay), this approach utilizes the multi-scale nature of the signal and therefore has better energy conservation properties, that is, the amplitude of the signal decreases less through denoising.

The multi-scale method used here is based on a time-invariant discrete orthogonal wavelet transformation (Nason and Silverman, 1995) because orthogonal wavelets can give the most compact representation of a signal. During the wavelet transform at each scale high- and low-pass filters are applied to the signal. The output from a high-pass filter is recorded as the wavelet coefficients and represents the details of the signal. The low-pass filter extracts the low-frequency components which are used in the next stage where another set of high- and low-pass filters is employed.

Wavelet shrinkage denoising does involve non-linear soft thresholding (shrinking) in the wavelet transform domain (Donoho, 1995). It is based on the assumption that wavelet coefficients of the true signal have high amplitude, opposed to the lowest magnitude coefficients that represent the noise (see Figure 3.3.5 for an example). Thus, by eliminating coefficients that are smaller than a predetermined threshold this noise can be removed. Summarized, it is a three step process: a linear forward wavelet transform, a non-linear shrinkage denoising of the resulting coefficients and a linear inverse wavelet transform. None of these steps needs a-priori parameterization of a particular model.

We now give a more formal definition. We assume we are given a noisy signal (the raw data) X consisting of m (noisy) samples

$$X(t) = S(t) + \epsilon(t), t = 1 \dots m$$

This raw signal X is assumed to contain the real signal S with additive noise ϵ. Let $\mathcal{W}(\cdot)$ denote the forward and $\mathcal{W}^{-1}(\cdot)$ the inverse wavelet transform operators and $\mathcal{D}_s(\cdot, \lambda)$ be the denoising operator with a data adaptive soft threshold λ. Our aim is to denoise X to estimate $\hat{S}$ which should be as close as possible to the original signal S. The three step process mentioned above is then:

- Linear forward wavelet transform: $Y = \mathcal{W}(X)$. This results in m noisy wavelet coefficients $y_{j,k}$.
- Non-linear shrinkage denoising, that is thresholding the wavelet coefficients which includes estimating the threshold $\lambda = \{\lambda_1, \dots, \lambda_j\}$ for each level j (see below)
  - $\lambda = d(Y)$
  - $Z = \mathcal{D}_s(Y, \lambda)$

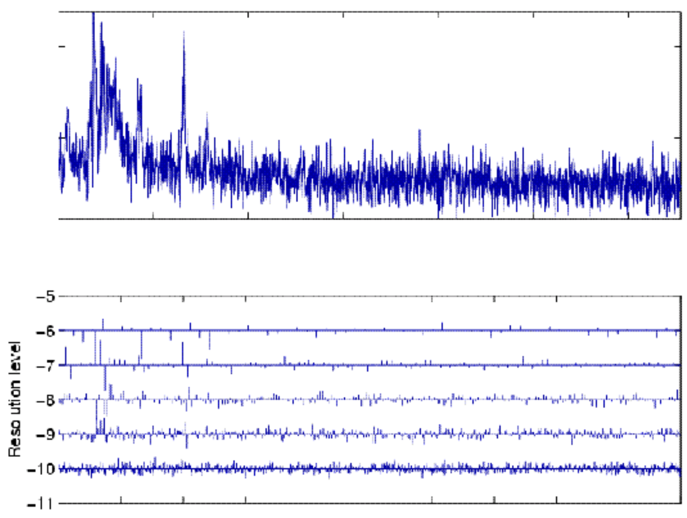


Figure 3.3.5: Noisy signal (top) and its discrete wavelet transform (using a Symmlet-10) at level $L = 6$. The wavelet coefficients are shown as spikes for the levels $(-6 \ldots -10)$. The size and direction a spike (coefficient) represent its magnitude and sign.

- Linear inverse wavelet transform: $\hat{S} = \mathcal{W}^{-1}(Z)$

Obviously, selecting the threshold λ is key to successful denoising. A global, non-adaptive λ to remove white noise is proposed by (Donoho, 1995):

$$\lambda = \sigma \cdot \sqrt{2 \cdot \log(m)}, \ \sigma = \frac{MAD}{0.6745}$$

where m is the length of the signal and MAD an estimator of the noise level determined by the median absolute deviation in the first scale (the constant 0.6745 makes the estimate unbiased for the normal distribution.).

From the tested shrinkage schemas (VisuShrink universal, Minimax, Stein's Unbiased Risk Estimate (SURE) and Minimum Description Length) the SURE approach (Stein, 1981; Donoho and Johnstone, 1995) was found to deliver the best results. It determines a threshold for each resolution level (scale) by the principle of minimizing the *Stein Unbiased Estimate of Risk*. This approach is smoothness-adaptive and has some interesting properties when used for the MALDI-TOF spectra that can contain spiky as well as smooth peaks: if the unknown signal S contains jumps, the reconstruction $\hat{S}$ does also and if S contains smooths regions $\hat{S}$ will be as smooth as the basis functions will allow. Figure 3.3.6 shows an example of this shrinkage applied to a usual MALDI-TOF spectrum.

Experiments have shown, that de-noising the signal improves classification accuracy of spectra (at the final stage of our pipeline, see section 3.8.5) by 4-5% on average, given all other parameters being equal. This is in good accordance with other studies, for example (Li et al., 2007) who achieve an improvement of 3-8% on similar SELDI-TOF serum data.

3.3.5 Baseline Correction

A baseline correction is performed to remove this rather low-frequency noise from the spectrum. Following (Breen et al., 2000; Sauve and Speed, 2004; Gröpl et al., 2005) we use a morphological TopHat filter (Zeng et al., 2006).

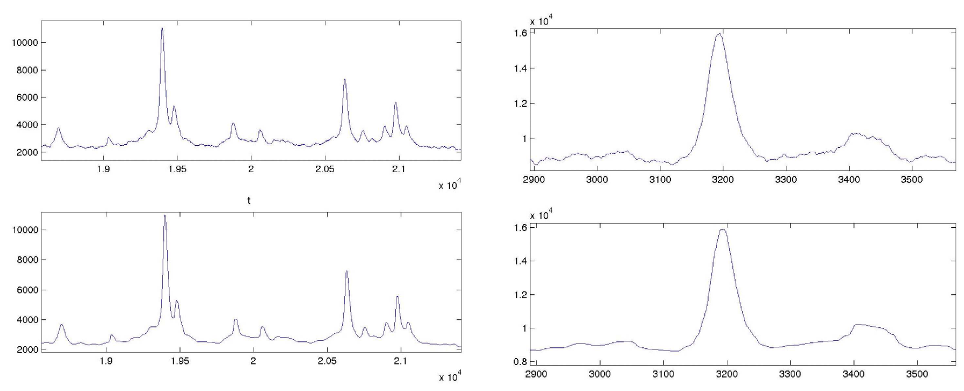


Figure 3.3.6: Application of wavelet-based denoising. The buttom row shows the de-noised version of the respective curve from the top row. Note that the curve structure is retained while the high frequent noise is removed.

Morphological signal analysis are useful for feature extraction, shape analysis and non-linear filtering. The TopHat transform function is a morphology function that allows the extraction of peaks and valleys in n-dimensional signals. An efficient TopHat transform was introduced by (Meyer, 1979) and is based on set concepts (Serra, 1982). Given a structural element Δ ("the base of the hat") and a (1d) signal X (e.g. a spectrum) the TopHat filter needs two basic morphological transformations of

Erosion

$$(X \ominus \Delta)(t) = inf\{X(t+i) : i \in \Delta\}$$

and *Dilation*

$$(X \oplus \Delta)(t) = sup\{X(t+i) : i \in \Delta\}$$

Using these the *Opening* function can be defined as:

$$X \circ \Delta = (X \ominus \Delta) \oplus \Delta$$

The actual TopHat operator is then defined as:

$$TH(X) = X - (X \circ \Delta)$$

Intuitively, during application of the TopHat operator, peaks that cannot contain Δ remain, while the others get eliminated.

Using this mathematical morphology analysis we can now eliminate certain spatial structures within the signal, in our case the baseline. Its simplicity and rapidity (mainly min/max operations) make it extremely handy for application to large amounts of data.

Fig. 3.3.7 illustrates this method. Note that for any $\Delta, X \geq X \circ \Delta$ everywhere; thus, $TH(X)$ is a non-negative signal opposed to other popular methods depending on polynomial fitting (Mazet et al., 2004), piecewise linear regression (Wagner et al., 2003) or convex hulls (Liu, Krishnapuram, Pratapa, Liao, Hartemink and Carin, 2003) do. This property will become extremely useful in the peak detection step (see section 3.4).

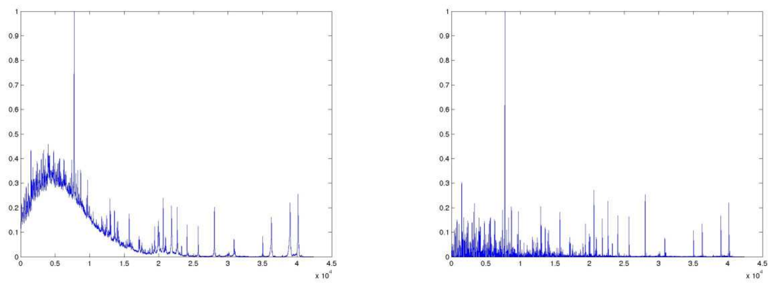


Figure 3.3.7: This shows the results of the TopHat Filter: the TopHat opening (see text) is subtracted from the raw signal (shown on the left) yielding the filtered version (on the right). Note that this method does not generate negative signals.

3.3.6 Normalization

Inter-spectrum normalization is the process of removing systematic variations between spectra. Many different techniques exist such as "Inverse Normalization" (Petricoin et al., 2002) or "Logarithmic Normalization" (Li et al., 2004). Our implementation follows the idea of the most frequently used method which is global normalization with respect to the average *total ion current* (TIC[2])[3] (Fung and Enderwick, 2002; Baggerly et al., 2003) with an important extension: from the set of spectra to be normalized all TIC values are computed, outliers removed and the remaining highest value (instead of the average) is used for the actual computation.

3.3.7 Computational Complexity Analysis of Preprocessing

The complexity of the single steps of the preprocessing

Denoising: $O(n \log n)$ (Besbeas et al., 2004)

Baseline reduction: $O(n)$ (Gao et al., 2003)

Normalization: $O(n)$

(n being the number of input samples) yields a total complexity of $O(n \log n)$.

[2]The TIC is the sum of the area of all peaks in a spectra.

[3]The normalization by the ratio $R = \frac{\text{“TIC of spectrum”}}{\text{“Average TIC of all spectra”}}$ is reported to be superior to other methods tested (Norris et al., 2005; Sauve and Speed, 2004).

3.4 Highly Sensitive Peak Detection

3.4.1 Introduction

In this (second) stage of the pipeline the detection of signals (*peaks*) in mass spectra is performed. Since only some peaks detected in this step have a biological meaning (that is, represent peptides) subsequent peak evaluation processes are crucial. These evaluations determine whether a given peak is just noise or actually represents a peptide. In later stages of the pipeline these *peptide peaks* are used to find differences between two groups of spectra (e.g. "diseased" vs. "healthy").

Recall that a spectrum consists of (x, y) value pairs that reflect the number of measured particles (y value) of a particular mass (x value). We define a peak as

Definition 3.4.1. Peak*: A set of successive x values ($x_i \dots x_j$) with corresponding y-values greater than zero where $y_{i-1} = 0$ and $y_{j+1} = 0$.*

In other words, all connected areas of a spectrum where the MALDI-TOF machine's detector did measure a signal are regarded as potential peaks.

The according step to peak detection in the Lego example is the detection of the bricks borders (see section 2.2). The same idea holds with the peak detection: a mass spectrum contains many peaks where we want to find start- and end-point (therefore the borders) of. In other words, the raw signal is scanned for regions that intersect the x-axis twice (begin and end) and start with positive slope. We call these regions *candidate peak* since we do not know yet whether or not they are real *peptide peaks*. Just as in the Lego example, we face the problem that if we cannot detect the borders reliably the algorithm might take two or more peaks for being one. This might be - for example - because

- the shape is often not clearly recognizable (noisy)
- peaks are convoluted (do overlay)
- noisy parts might look - just by chance - like a peak
- peaks might be very small, that is, below the noise level

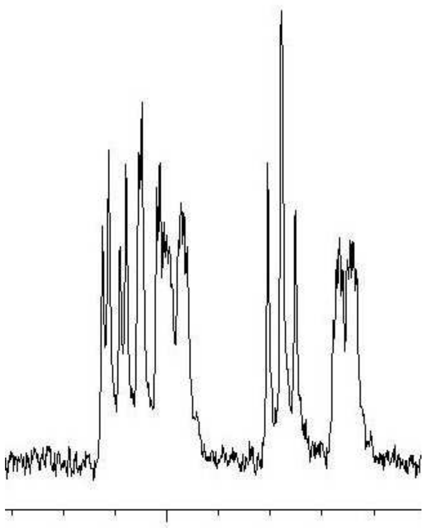


Figure 3.4.8: A zoom into a common raw spectrum with some clearly identifyable large peaks inbetween many small peaks usually regarded as noise. We left out the scale since we want to draw the attention to these two scales (noise vs. signal).

The basic approach to detect these errors is to compare the candidate peaks with a previously learned model. In other words, if we know how a peak shape should look like, we can check if a candidate peak is valid, or we missed a border, or just detected noise.

The key assumption we use in our algorithm is that most peaks consist of Gaussian-like shapes (sub-peaks). To understand why this is a good model, let us recall the functioning of a mass spectrometer and some chemical basics. If we were in a perfect world, each molecule in a sample would have a well-defined mass and represented as a very thin peak at exactly this mass. So why do we see a Gaussian-like peak ? The first - simple - reason is the inaccuracy of the measurement process, since (imprecise) time-of-flight data is converted into (molecule) mass and small errors can occur. This leads to small shifts in x direction and broadens the peak. Secondly, due to the existence of isotopes [4] , the more complex a molecule is the more different varieties, with respect to mass, exist. For example, oxygen occurs in nature as three different (stable) isotopes: ^{16}O (99.765%; 8 protons, 8 neutrons), followed by the rare isotope ^{18}O (0.1995%; 8 protons, 10 neutrons) and the even rarer isotope ^{17}O (0.0355%; 8 protons, 9 neutrons). Obviously, the more complex the molecule, the more combinations of isotopes (and hence masses) are possible. Since some combinations are more likely than others, the isotopes are independent of each other, and there is usually a high number of a particular molecule we see a Gaussian-like shape (Central limit theorem). Depending on the type of machine used this shape can be resolved in its *isotopic components* (see Figure 3.4.9).

The knowledge of the isotope distributions enables us to exactly calculate the shape and position a molecules peak will (should) have. So if we find a peak-like shape at a certain position we can determine the similarity to the calculated shape and accept this shape or perform further analysis.

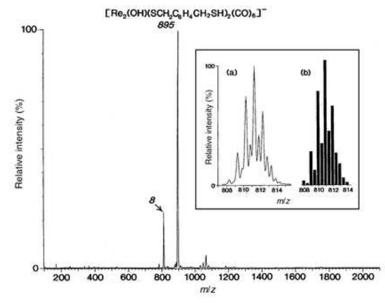


Figure 3.4.9: Sample Spectrum. The inset shows a comparison of (a) experimental and (b) calculated isotope distribution patterns for the peak at m/z 811.

3.4.2 Common Approaches

Almost all peak detection algorithms rely on the shape comparison technique described above. They usually differ in how they detect candidate peaks. What they have in common is the usage of threshold driven detection techniques. That is, each candidate peak must be higher than a predetermined signal-to-noise threshold depending on the calculated noise level (see e.g. (McDonough and Whale, 1995)).

Drawbacks of common Approaches

As shown exemplarily in Fig. 3.4.10, by assuming a noise level of 50[5] and using a signal-to-noise ratio of 3[6] about 85% of the 1332 potential (candidate) peaks in this particular spectrum would be discarded and their assigned information lost. Although most of these peaks essentially *are* noise, some might carry important information. This means, that these artificially introduced barriers would prevent detection of small signals in a very early pre-processing stage.

The subsequent sections describe our new approaches to overcome this signal-to-noise barrier, that means increasing sensitivity without decreasing specificity.

3.4.3 Our Approach

To avoid loss of potentially important information by not considering (small) peaks in the preprocessing we take the most simple solution and regard everything as a *candidate peak* that has a start point $P_{i,s} \in S$ and an end point $P_{i,e} \in S$, $S = s_2 \ldots s_n$ being the set of n points defining a spectrum. Then the tuple $(P_{i,s}, P_{i,e})$ defines the i*th* candidate peak ranging from $S_s \ldots S_e$. The requirements for these points to meet are:

[4] Atoms with the same number of electrons and protons, but different numbers of neutrons, are called isotopes. Different isotopes belong to the same element because they have the same number of electrons, which means that they all behave almost the same in chemical reactions. It was discovered during the Second World War that isotopes of the same element can be separated by physical and chemical methods.

[5] different noise-estimators compute values ranging from 50 to 150

[6] a commonly used value to get reliable results

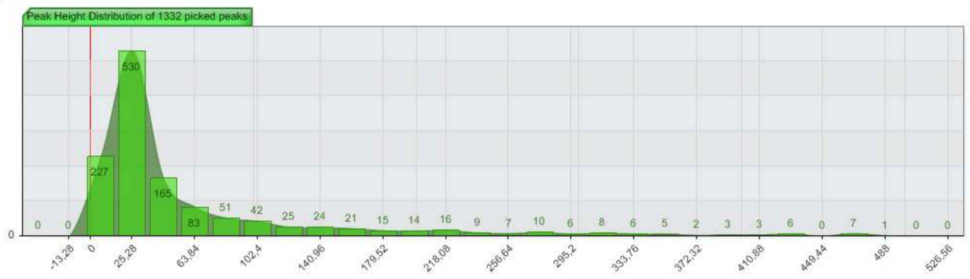


Figure 3.4.10: Histogram of peak heights from a randomly chosen spectrum. Note that only peaks smaller than 500 are displayed. The curve in darker green is a spline based approximation to the top mid-points of the histogram bars.

- $x(P_{i,s}) < x(P_{i,e})$ - the end point's x value must be greater than the start point's x value.

- $y(P_{i,s}) = 0$ - the y value of the start point must equal to zero.

- $y(P_{i,e}) = 0$ - the y value of the end point must equal to zero.

- $\forall S_k \in \{S_{x(P_{i,s})} \ldots S_{x(P_{i,e})} : S_k > 0\}$ - the y values of all points between the start and end point must be greater than zero.

Where $x(\cdot)$ returns the x and $y(\cdot)$ returns the y value of a given spectrum point. Thus, we are scanning through a spectrum S looking for intersections of the data curve with the x-axis. Everything between every two intersection we call a *candidate peak*.

However, most of these candidate peaks found with this simple algorithm are not actually real peaks. Most are overlaying single peaks either resulting from poor resolution of the MALDI-TOF machine, because of isotopic patterns, or highly complex peptide mixtures.

Pseudo Code

Algorithm 2 Peak Detection

```
Require: Preprocessed Spectrum $(x_i, y_i) == s_i \in S$ as sequence of value
  pairs, $i = 1 \ldots n$
  {Look for start of first peak}
  $i \leftarrow 0$
  while $y(i) > 0$ do
    $i \leftarrow i + 1$
  end while
  candidatepeak_start $\leftarrow i$
  {Scan through value pairs}
  for $i =$ candidatepeak_start $\ldots n$ do
    if $y(i) == 0$ then
      if $s_i$ is start of candidatepeak then
        candidatepeak_start $\leftarrow i$
      else {$s_i$ is end of candidatepeak}
        candidatepeak_end $\leftarrow i$
        $RP \leftarrow$ candidatepeak area from candidatepeak_start to candidate-
        peak_end
        { Deconvolve $RP$ to set of real peaks }
        $P \leftarrow$ sub-peaks of $RP$
        { De-isotope }
        if $P$ is isotopic pattern at mass $x(i)$ then
          Determine parameters of (single peak) $P$
          $PL \leftarrow PL + P$
        else
          for all $p \in P$ do
            Determine parameters of (single peak) $p$
            $PL \leftarrow PL + p$
          end for
        end if
      end if
    end if
  end for
  return List of peaks $PL$
```

Detection of Candidate Peaks

The initial peak detection simply determines the location of potential peaks, a process often referred to as *seeding*. Utilizing the properties of the TopHat filter (see Section 3.3.5) it is sufficient to detect interception points of the curve with the X-axis. These points define start- and end points of potential peaks and are stored in a database for further analyses.

The advantage of this approach is that even smallest peaks are considered for consecutive steps. However, a deliberate validation algorithm must be applied to this set of candidate peaks to distinguish real peaks from noise and detect and deconvolute overlapping peaks.

Analyzing Candidate Peaks

In mass spectra of complex protein mixtures (such as serum) most of the peaks detected are broadened and/or highly convoluted. This results for example from molecular fragments having very similar masses and thus partly overlay, from poor machine resolution, or different isotopic forms of the same molecule. Therefore, a successful peak detection algorithm needs to deconvolute those peaks. A widely accepted method assumes a blurred peak to be a mixture of Gaussians and tries to resolve it back into its original components. A commonly used technique is *Maximum Entropy* that has been originally developed for clarifying blurred images (see (Guiasu and Shenitzer, 1985)). Based on this idea we have developed an approach to separate and evaluate the assumed mixture of Gaussians. The key steps for each candidate peak found are as follows:

1. Determine number of Gaussian components by density estimation using the *Greedy Expectation Maximization* algorithm (Verbeek et al., 2003). This algorithm has been shown to have a very good performance even on large mixtures often found in peaks at higher masses ($>$ 3000Da). (For a comparative study see (Paalanen et al., 2005).)

2. To account for isotopic forms a de-isotoping step is carried out by fitting a mass dependent pre-calculated model (see (Zhang et al., 2005) for details) if more than one Gaussian component is found. If successful, the peaks involved are tagged as belonging to the same molecule(-fragment). If the quality of the fit is too poor the peak is split according to the number of Gaussians found and for each of the new parts step 1 is performed again.

3. Determine and store the parameters (height, width, center, area, shape quality[7]) of this peak.

3.4.4 Computational Complexity Analysis of Peak Picking

The complexity of the peak picking is $O(n)$ where n is the number of input points since we loop over all points and determine properties of some areas (candidate peaks) which are performed in constant time.

3.4.5 Comparison to Threshold-driven Algorithms

To obtain a first proof-of-principle we spiked a subset of human serum samples[8] (see section 4.1.2) with a peptide mix[9]. We split 16 different samples into five groups each. Before sample pretreatment and measurement each of the groups was spiked with one of the following concentrations: 121.21nMol/L, 0.76nMol/L, 0.30nMol/L, 3.03pMol/L, 0.075pMol/L, resulting in 320 spectra (64 for each concentration group due to 4-fold spotting).

[7]This is achieved by geometrical hashing (Wolfson and Rigoutsos, 1997). The hashing algorithm returns a discrete value $c \in \{0, 1 \dots 5\}$, indicating the class the hashing algorithm has assigned to this shape. $c = 0$ means noise and $c = 1..5$ peak, where $c = 5$ is assigned if the peak looks perfect. The categories are trained a priori.

[8]The protocol used for preprocessing and (magnetic bead) fractionation has been described in (Baumann et al., 2005).

[9]Protein calibration standard mix (Part No.: 206355 & 206196 purchased from Bruker Daltronics, Leipzig, Germany)

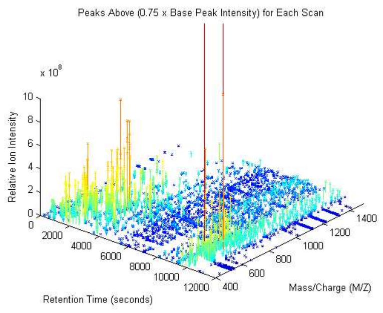
Substance	**Plat-form**[3]	**Theor. Center**	**Center found** (Concentration of spiked peptide mix as stated below)				
			121.21nMol/L	0.76nMol/L	0.30nMol/L	3.03pMol/L	0.075pMol/L
Angiotensin II	P.NET	1047.20	1047.1	-[1]	-[1]	-[1]	-[1]
Angiotensin II	CPT	1047.20	1046.9	-[2]	-[2]	-[2]	-[2]
Angiotensin I	P.NET	1297.51	1297.6	1298.0	1299.3	1299.2	-[1]
Angiotensin I	CPT	1297.51	1297.2	-[2]	-[2]	-[2]	-[2]
Bombesin	P.NET	1620.88	1620.9	1618.1	1617.2	1617.2	-[1]
Bombesin	CPT	1620.88	1620.6	-[2]	-[2]	-[2]	-[2]
ACTH clip 18-39	P.NET	2466.73	2466.8	2465.8	2465.8	2466.2	-[1]
ACTH clip 18-39	CPT	2466.73	2466.2	-[2]	2466.1	2465.9	-[2]
Somatostatin 28	P.NET	3149.61	3149.5	-[1]	-[1]	-[1]	-[1]
Somatostatin 28	CPT	3149.61	3149.0	-[2]	-[2]	-[2]	-[2]
Insulin	P.NET	5734.56	5734.3	-[1]	-[1]	-[1]	-[1]
Insulin	CPT	5734.56	5734.2	-[2]	-[2]	-[2]	-[2]

Table 3.4.1: Results of the spiking experiments (see text for explanation). Note that no calibration has been performed for the Proteomics.NET platform.
[1]: No significant masterpeak in this range at this concentration found, [2]: No significant peaks in this range at this concentration found, [3]: CPT: Bruker ClinprotTools, P.NET: Our Proposed Platform: Proteomics.NET

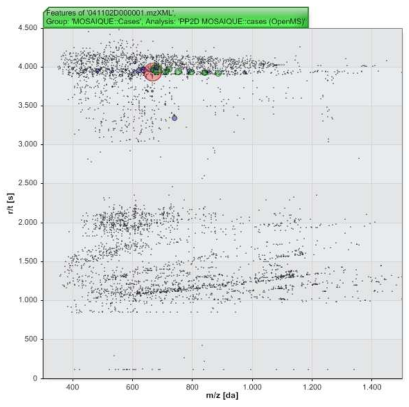
We then processed each resulting raw spectrum as described above. For each of the five resulting concentration groups we ran the peak detection algorithm of our platform and of the Bruker software. Table 3.4.1 summarizes the findings:

The algorithms successfully detect peaks even for very small concentrations at pMol/L level. This is exemplarily shown for the hormones Angiotensin, Bombesin and ACTH clip 18-39 which can be detected in a very low and biologically relevant concentration range ($\sim$3 pMol/L). Peaks for Angiotensin and Bombesin are not detected by commercial software[10]. Therefore, in these examples, our algorithm is at least 20000 times more sensitive than a commercial algorithm using a signal-to-noise threshold.

[10]ClinProTools 2.0, Bruker Daltronics. Parameters used: Signal-to-Noise Level: 3, Peak Detection Algorithm: Centroid



(a) Raw 2D spectrum



(b) Heat map of 2D spectrum.

Figure 3.5.11: LC/MS 2D map example. Left: This view exhibits the single 1D (m/z vs. intensity) spectra. The y-axis shows the retention time. Right: 2D spectrum from above showing m/z vs. retention time. The peak intensity is coded in the circle radii.

3.5 Peak Detection in 2D Maps

This section introduces our peak detection algorithm for 2D spectrum maps. A 2D map is created from (hundreds of) single 1D spectra which are arranged successively (ordered by acquisition time). These single spectra usually result from one MS run preceded by some fractionation technique, such as liquid chromatography (LC, see section 2.1.1). Therefore, the m/z and intensity (x and y) axes stay the same and a new z-axis is introduced denoting the *retention time* of a peptide (peak). The retention time states the elapsed time since the beginning of the MS experiment at that a particular spectrum is acquired. This means, a spectrum S_t taken at time t contains all peptides that were eluted from the column (in the LC/MS case) between time $(t-1)$ (when the previous spectrum was acquired) and t.

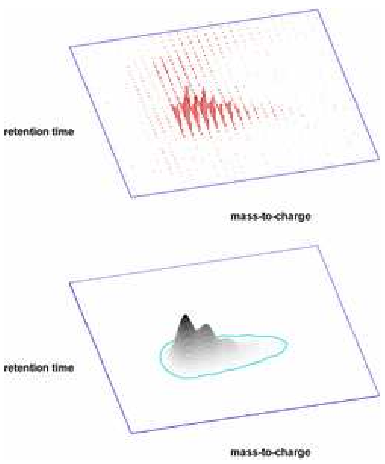


Figure 3.5.12: LC/MS 2D Feature. (Picture taken from (OpenMS, 2008).) The top image shows the peaks of the 1D spectra this feature was extracted from. The bottom figure shows a 2D function fitted on the 1D peaks.

Figure 3.11(a) shows such a 2D map: the single spectra are plotted on the m/z (x) axis, ordered by increasing retention time; the intensities are color coded from dark blue (very small peak) to dark red (very high peak). Taking a look from above (thus neglecting the intensity axis) yields Figure 3.11(b): here, the 2-dimensional structure of a peak gets obvious (shown more clearly in Figure 3.5.12), resulting from similar peaks (with respect to height) from consecutive spectra.

The 2D peak detection algorithm is based on the 1D peak detection: first the peak detection is performed on the single 1D spectra of a 2D map. This process is easily distributable to many working machines or processors and can be run in parallel (see chapter 5). The peak detection of the 1D spectra results in peaks (isotope patterns) for each single spectrum. Subsequently, the 2D-detection is performed which consists of the following key steps:

- Load picked peaks for all 1D spectra of the 2D Map as 2D points with attached attributes (such as $(\frac{m}{z}, t)$-position, parameterization of isotope pattern, goodness of isotope-pattern fit, etc.).

- Create 2D (orthogonal) range tree (de Berg et al., 2000) which needs $O(n \log n)$ time and space for creation and storage[11], respectively, and can answer range queries in $O((\log n)^2 + k)$[12] (k being the number of results). Since in a typical (medium resolution) map $n \sim 2.000.000$, a query needs about $(36 + k)$ comparisons in time and 12MB in space. Of course, the analysis could also be performed directly by querying the database but by using range trees this can be done on a remote worker (see section 5.4) without the need for using the potentially slow database connection.

- For each spectrum S_t (sorted increasingly by retention time t): If an yet unprocessed peak is found at position $(\frac{m}{z}, t)$

  – Use this peak as seed and extend a bounding box around it.

    ∗ The extension in *m/z (x) direction* is given by the length of the isotope pattern + 10% (that is, if an isotope pattern spans from 1000 to 1010da the box would have the x-dimension: 999 to 1011da).

    ∗ In *retention time (y) direction* (successively $(t - i)$ and $(t + i)$) the extension is done as follows: if S_i is the current 1D spectrum, get the peaks of the next spectrum $(S_j = S_{i\pm1})$ within the determined x range. If the peaks found are similar (see below) to the peaks of S_i this step is repeated until no further extension is possible.
    If the peaks found are not similar the next two spectra ($S_k = S_{i\pm2}$ and $S_l = S_{i\pm3}$) are checked as well to account for missing data. If the peaks of S_k or S_l are similar to the peak of S_i the respective spectrum is set as the current one and this step started all over.

  – Fit 2D isotope pattern (mixture of 2D Gaussians) on peaks found within the bounding box. For each 2D Gaussian[13] we can compute the *center of mass* along the retention time axis and the m/z spread along the m/z axis for each gaussian from the 1D isotopic patterns (consisting of a mixture of Gaussians).

  – Mark used peaks as processed

This procedure results in a list of 2D peaks parameterized by a mixture of 2D Gaussians.

3.5.1 Similarity Measures for Curves

The similarity (or distance) $d(A, B)$ of two curves A, B (e.g. isotope patterns modeled as mixture of 1D Gaussians) is measured by the difference of their curvature based *Turning Function* Θ_A (Arkin et al., 1991) (see (Veltkamp and Hagedoorn, 2000) for a discussion of differently similarity measures) that is well suited to capture differences in isotope pattern shape but is not sensitive to height differences.

[11]Optimal storage of $O(\frac{n \log n}{\log \log n})$ is possible by using R-trees.

[12]Or $O((\log n) + k)$ in an improved *fractional cascading* version.

[13]$f(x, y) = A \cdot exp(-(\frac{(x-x_o)^2}{2\sigma_x^2}) - (\frac{(y-y_o)^2}{2\sigma_y^2}))$ where A is the amplitude, x_0, y_0 the center and σ_x, σ_y the x and y spreads.

3.6 Peak Registration (Alignment)

The term *registration* is borrowed from the field of computer vision. Here, an object is often sampled from different perspectives and is therefore in different coordinate systems. Image registration is the process of transforming these into one global coordinate system. Obviously registration is necessary to enable reliable comparisons or integrations.

As introduced in the Lego example (see section 2.2), to enable detection of differences between two classes we first need to group together the same items (bricks) of each individual (picture) and subsequently detect sub-groups (differences) in these item groups. As we grouped together Lego bricks in the example we will now group peaks in our spectra world. The problems arising are: (a), similar to the Lego example, the positions of the peaks are a little fuzzy, and (b) a spectrum might be linearly shifted and/or non-linearly transformed. Therefore, we have basically the same problem as in computer vision: we first need to transform the elements (peaks) of all spectra into one *global* system.

In the previous section we showed how to detect and evaluate peaks in single spectra. The question we now pose is: if we find a peak in a spectrum at position X having properties Y_i - do there exist peaks at the same position and with similar properties in other spectra of the same group ? The basic assumption underlying this question is that a peak not occurring in all (most) spectra of a particular group is either specific to the individual or it is just noise. In the following, we will call a group of peaks from different spectra that occur at the same (x-axis) position a *Masterpeak* (see Figure 3.6.13 for an example). The important thing to note here is that we group the peaks and operate on the level of peak groups. We therefore leave out the noise in between and do not operate on a spectrum level.

3.6.1 Our Approach

In order to identify a particular peak across spectra a list of so-called *masterpeaks* is maintained per spectra group (e.g. male or healthy). A masterpeak comprises peaks having similar properties (m/z value, height, shape, etc.) across spectra in this group. From the comprehensive distribution of property values the real values for a masterpeak will be derived in later stages.

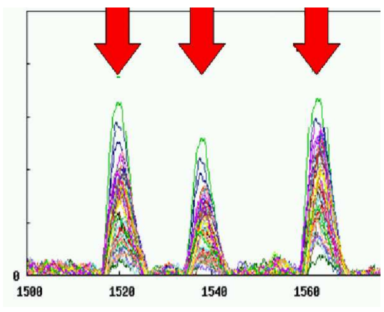


Figure 3.6.13: This shows three masterpeaks (indicated by red arrows). A masterpeak is a *cluster of peaks* occurring at the same position in different spectra of the same group.

Pseudocode

Key Steps:

1. Create candidate clusters
2. Refine clusters (Bayesian Clustering)
3. Determine cluster properties
4. Merge similar clusters
5. Result: List of masterpeaks

Algorithm 3 Peak Registration (Finding Candidate Masterpeaks)

```
Require: Peak list $PL$ of spectra group (e.g. healthy) sorted by peak centers
  {Create candidate clusters (CC)}
  {Alternatively, hierarchical clustering could be performed}
  $CC \leftarrow \emptyset$
  $curC \leftarrow PL(0)$
  for $i = 1 \dots |PL|$ do
    $p1 \leftarrow PL(i-1)$
    $p2 \leftarrow PL(i)$
    if $p1$ and $p2$ overlap partly then
      $curC \leftarrow curC + p2$
    else { $p2$ does not overlap with $p1 \rightarrow$ End of Cluster found}
      if $|curC| > 1$ then
        $CC \leftarrow CC + curC$
      end if
      $curC \leftarrow p2$
    end if
  end for
  {Refine candidate clusters}
  $C \leftarrow \emptyset$
  for all $curC \in CC$ do
    $C \leftarrow C + CRP(curC)^a$
  end for
  {Determine cluster properties}
  $MPs \leftarrow \emptyset$
  for all $curC \in CC$ do
    Determine set of properties $P$ (center, height, ...) of $curC$
    $MPs \leftarrow MPs + (curC, P)$
  end for
  {Merge similar clusters}
  $MPs_{merged} \leftarrow \emptyset$
  $i \leftarrow 0$
  while $i < |MPs| - 1$ do
    $curMP \leftarrow MPs(i)$
    $j \leftarrow i + 1$
    while $(j < |MPs|)$ and $(MPs(i)$ is similar to $MPs(j))$ do
      $j \leftarrow j + 1$
      $curMP \leftarrow$ merge $(curMP, MPs(j))$
    end while
    $MPs_{merged} \leftarrow MPs_{merged} + curMP$
    $i \leftarrow j + 1$
  end while
  return List of masterpeaks $MP_{merged}$
```

[a]: Result of Bayesian Clustering (Chinese Restaurant Process, CRP)

Finding Candidate Masterpeaks

To build a set of potential masterpeaks the following two steps are carried out:

1. Center and width of every peak identified in step 3.4.3 in a group of spectra under scrutiny (e.g. healthy or female) are stored in a temporary

table, ordered by their center.

2. Candidate clusters of these peaks are built with respect to the centers and the width of these peaks. Peaks belong to the same cluster if they overlap in at least one point. Since we can simply scan through this ordered set of peaks with a linear number of comparisons, the complexity is $\mathcal{O}(n)$. Alternatively, complete-linkage hierarchical clustering could be performed (Tibshirani et al., 2004) to build the clusters which is computationally more expensive.

Refining the Clusters

We now have a set of candidate clusters often containing more than one *real* group of similar peaks. This step is going to resolve these groups by a Bayesian Clustering approach. From the clusters found in this step all properties such as center, height or width are derived. From the law of large numbers we know that the average values will converge to the real values. The probabilistic object that underlies this approach is a distribution on partitions of integers[14] known as the (weighted) *Chinese restaurant process* (CRP) (Aldous, 1983; Ishwaran and James, 2003; Lo, 2005).

The CRP can be best described by a process where N customers (here: peaks) sit down in a Chinese restaurant with an infinite number of tables $C_1, C_2, \ldots$ and each table has an infinite number of seats. Suppose customers X_i arrive sequentially. Per definition the first guest x_1 sits down at the first table. The $n+1$th subsequent customer x_{n+1} sits at a table drawn from the following distribution:

$$\begin{aligned} p(\text{occupied table } i \mid \text{previous customers}) &= \tfrac{|C_i|}{\alpha+n} \cdot R \cdot s(x_{n+1}, C_i) \\ p(\text{new table} \mid \text{previous customers}) &= \tfrac{\alpha}{\alpha+n} \end{aligned}$$

where $|C_i|$ is the number of customers already sitting at table C_i, R is a rescaling factor and $\alpha > 0$ is a parameter defining the CRP. Obviously, the choice of the similarity function $s(\cdot)$ is crucial and is explained in the following paragraphs.

Let $\mathcal{D}_{A,C_i}$ be the distribution of a property A (e.g. center or height) of the peaks "sitting" at table C_i. (For example, $\mathcal{D}_{\text{center},C_2}$ would be the distribution of peak centers of all peaks "sitting" at the 2^{nd} table.) Then, $\mu(\mathcal{D}_{A,C_i})$ is the mean of that distribution. Let $A(x_j)$ be the value of property A of peak j. (For example, center(x_j) would return the center of peak x_j.) $s(\cdot)$ has the following properties:

1. The distance of the center of a peak to the average center of an existing group of peaks cannot be further away than 2 Da.

2. $s(x_j, C_i)$ is the likelihood of peak x_j belonging to table C_i depending on how similar the properties of x_j are to the peaks already at table C_i.

This results in:

$$s(x_j, C_i) = \begin{cases} 0 & \text{if } \mid \mu(\mathcal{D}_{\text{center},C_i}) - \text{center}(x_j) \mid > 2 \\ \sum_{A \in \mathcal{PP}} \mathcal{D}_{A,C_i}\big(A(x_j)\big) & \text{otherwise} \end{cases}$$

[14]Interestingly, the partition after N steps has the same structure as draws from a Dirichlet process (Ferguson, 1973; Blackwell and MacQueen, 1973).

$\mathcal{PP}$ being the set of peak properties. $\mathcal{D}_A(C_i)$ is constructed as follows: Kernel Density Estimation (KDE)[15] (Scheather, 2004) is performed on $A(x_k)\ \forall\ x_k \in C_i$, interpolated and scaled to result in a probability density function.

We have used this particular clustering approach since preliminary experiments have shown that this technique was able to best resolve the clusters in the $|\mathcal{PP}|$-dimensional space we work in (typically $|\mathcal{PP}| = 6 \ldots 8$). Analyses have shown that the $\mathcal{D}_A(C_i)$ are usually not unimodal thus favoring a clustering schema that does not create spherical clusters, such as k-means clustering (see e.g. (Deonia et al., 2007)).

Determine Cluster Properties

This step determines the properties (such as center, height, ...) of the clusters found in the previous step. For each property a KDE k is performed on the values of the single peaks the masterpeak consists of. If k is not similar to a normal distribution (tested by: Kolmogorov-Smirnov test, Lilliefors test, Anderson-Darling test, Ryan-Joiner test, Shapiro-Wilk test, Normal probability plot (rankit plot), Normality test, Jarque-Bera test) the masterpeak is flagged.

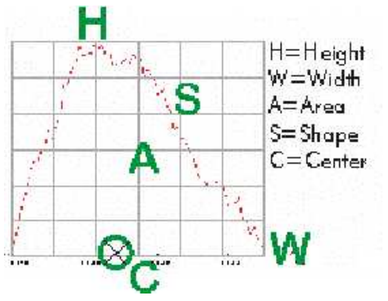


Figure 3.6.14: These are the parameters being determined for each peak.

Merge Similar Clusters

Since the clustering in the previous step is not deterministic this step repairs clusters that have been divided into two or more sub-clusters. Two clusters are merged together if all properties or all properties except the center are similar (measured by the Jensen-Shannon divergence, see section 3.7.1).

Outcome: List of Masterpeaks

The above procedure finally results in a list of masterpeaks. That is a list of peaks clustered by positions and properties (such as position or height) represented by the average values for these properties.

3.6.2 Other Approaches

An alternative approach to enable different spectra for comparison is to align them and described below. This means, define some reference key peaks (e.g. based on known house-keeping molecules), find these peaks in each spectrum and reorientate each spectrum towards these peaks. Obvious problems are:

- How to detect the key peaks ?
- What if key peaks are missing ?
- What if there is a peak similar and next to a key peak ? Which is the right one ?

The main issue here is that during the reorientation process a spectrum gets partly distorted, because between two key peaks a linear adjustment takes place but the parts are not directly tied together.

This is shown exemplarily in Figure 3.6.15: spectrum (b) is reorientated on the basis of its detected key peaks towards the position of the reference key

[15]Following the Parzen Window approach with Gaussian Kernel.

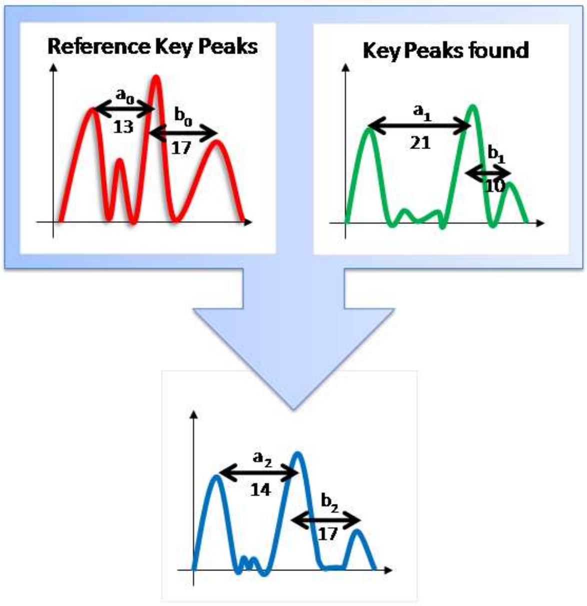


Figure 3.6.15: This shows a spectrum alignment using key reference key peaks. The top left picture show the reference key peaks and their respective distances in x-direction. The top right picture show a new spectrum to be aligned with the reference. The result of this alignment process is shown in the bottom picture.

peaks. As can be seen, the part in range a_1 is compressed while range b_1 is stretched and partly filled with zero elements.

Other Alignment Algorithms

The problem of spectra alignment appears not only in mass spectrometry but for many other data, such as in NMR spectra alignment (see (Kim et al., 2006) for an Bayesian approach) or electrophoretic lane alignment (see (Aittokallio et al., 2001) for a dynamic time warping approach). In all of these domains (and many more) different and similar algorithms have been developed (see e.g. (Mäkinen, 2007) for an example derived from sequence alignment). One large family of algorithms we have analyzed during this work have the following drawbacks:

- They are operating globally, that is, local changes (during alignment) of an ordered spectrum $x_0 \ldots x_n$ at a particular position x_i affects all positions $x_{i+1} \ldots x_n$.
- They align exactly two spectra, that is, a new spectrum to a reference spectrum.
- They can not incorporate any statistical information.

As we have shown above, our approach allows to circumvent these shortcomings while using statistical information which are incorporated naturally by aligning groups of peaks (masterpeaks) opposed to single peaks from two single spectra.

3.6.3 Computational Complexity Analysis of Peak Registration

The complexity analysis of these steps yields:

Sorting the peak list: $O(n \log n)$ (e.g. by quicksort)

Finding the cluster: $O(n)$

Refining the clusters: $O(n)$

Determine cluster properties: $O(n)$

Merge clusters: $O(n)$

so we have a total complexity of $O(n \log n)$.

3.7 Identifying Potential Features

This step identifies masterpeak (potential features) that can be used to discriminate two groups based on their respective properties (e.g. differences in average height, see Fig. 3.6.13). That is, we consider a masterpeak a feature if this masterpeak can be used to discriminate two sets of spectra. For example, if a masterpeak at position x does only occur in one of these spectra sets it is a feature since the detection or absence of this peak would clearly assign it to one of the groups.

3.7.1 Our Approach

After the preprocessing steps we now have information about masterpeaks of two patient (spectra) groups under scrutiny. To enable the creation of fingerprints (see next section 3.8) we first need to create a set of potential differences between these two groups of spectra. We define two spectra to be different (with respect to one particular property) if

a) a masterpeak existent in one group does not occur in the other group

b) a masterpeak exists in both groups but differs significantly in some property between the two groups.

In other words, the feature detection step identifies a set of masterpeaks that differ significantly in particular properties (e.g. height, width) between two groups of spectra with respect to some metric. With these information we can subsequently analyze for sub-sets / patterns (fingerprints) by detection and subsequent selection of the most significant combination of features.

Choosing the Metric

A metric or distance function defines a distance between two elements of a set. The elements of our set are masterpeaks that are defined by property distributions of their assigned single peaks, such as m/z values, height or area. What we want is a distance function that equals to some very large number (or infinity) if it does not make sense to compare them (that is, their respective m/z values are too different) or incorporates the (dis-)similarity of their property distributions otherwise.

Therefore, we need some (symmetric) method of measuring the similarity between two probability distributions which we found in the Jensen-Shannon (JS) divergence (see e.g. (Gómez-Lopera et al., 2000) and references therein) because it can be computed quickly, has shown good results in similar applications and does not assume strong properties of the data, such as being normally distributed: For probability distributions "P" and "Q" of a discrete variable the JS-divergence of "Q" from "P" is defined as:

Definition 3.7.1. *Kullback-Leibler (KL) divergence (S. Kullback, 1951):*
$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$.

Definition 3.7.2. *Jensen-Shannon (JS) divergence (Lin, 1991):*
$D_{JS} = \frac{1}{2}\left(D_{\mathrm{KL}}\left(P\middle\|\frac{P+Q}{2}\right) + D_{\mathrm{KL}}\left(Q\middle\|\frac{P+Q}{2}\right)\right)$.

Of course there are other probability distance measures, for example histogram intersection (Jia et al., 2006), Kolmogorov-Smirnov distance (Fasano and Franceschini, 1987) or the earth mover's distance (Rubner et al., 2000), but these usually have quite strong requirements to the data.

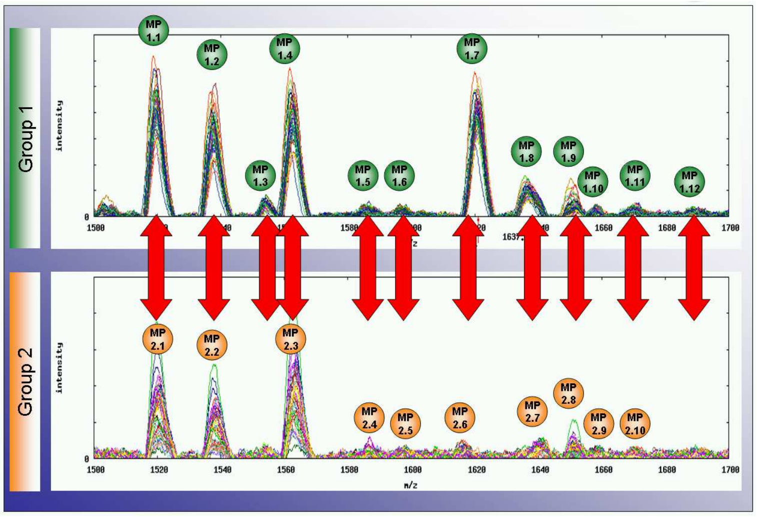


Figure 3.7.16: This shows identified masterpeaks of two groups (top: men; bottom: women). The alignment process assigns pairs of masterpeaks having similar properties (such as m/z value). These assignments are indicated by the red arrows.

Algorithm

From the preprocessing steps we have acquired a list of masterpeaks of groups G_1, G_2 (e.g. cancer group vs. healthy group). To obtain a set of masterpeaks that differ in some defined property (e.g. average height) we perform the following key steps:

i) Alignment of Masterpeaks in G_1, G_2 (masterpeak alignment across groups)

ii) Calculation of aligned Masterpeak Pair (MPP) property Jensen-Shannon (JS) differences

iii) Order this list by distances.

This yields a list of pairs of aligned masterpeaks of G_1 and G_2, ordered by their respective distances.

Pseudocode

The basic process is as follows:

Algorithm 4 Extracting features

```
Require: Lists of masterpeaks of group 1 & 2, respectively.
  {Masterpeak Alignment}
  Match the masterpeaks of group 1 & 2 by algos described in section 3.7.1.
  {Calculation of Masterpeak Property Differences}
  return A sorted list containing tuples of aligned pairs of masterpeaks with
  their respective distances.
```

Preprocessing 1: Masterpeak Alignment (Across Groups)

The majority of masterpeaks obtained by the preprocessing steps occurs in each group of spectra (e.g. S_1, S_2) at the same position and having identical heights, due to the almost identical blood proteome of two humans. Remember that a masterpeak is a set of peaks where each single peak stems from a distinct spectrum and represents a molecule (or peptide) of a certain weight. If we now find a masterpeak in two different groups at the same (m/z) position we have most likely found the same molecule (peptide) in the two groups. If these masterpeaks differ significantly in height we could differentiate by them. However, if the two masterpeaks have similar height we call them *non-informative* and want them to be tagged as such.

Following this, we first have to match masterpeaks of two groups occurring at the same m/z position. Obviously, sometimes a masterpeak from group S_1 cannot be matched with a masterpeak from group S_2 because this molecule (peptide) is just not present in S_2. This process is called *Masterpeak Alignment.*

We have implemented two different approaches to solve this problem, a naive version and an implementation as an *Minimum Weight Maximum Cardinality Bipartite Matching* formulated as *Linear Program* and solved by the Munkres (Hungarian) Algorithm which are compared in section 3.7.3.

Approach 1: Naive Solution

Algorithm 5 Masterpeak Assignment - Naive

```
Require: Lists of masterpeaks $MP_1, MP_2$ of groups 1 & 2 respectively
  while $MP_1$ has more elements do
    $MP1_{cur} \leftarrow$ next element from $MP_1$
    $MP2_{candidates} : \{ s \mid \text{m/z}(s) - \text{m/z}(MP_1)| \leq 2\}$
    if $|MP2_{candidates}| = 0$ then
      insert $(MP1_{cur}, \emptyset)$ into $L_{MP\_PAIRS}$
    else
      for all $p \in MP2_{candidates}$ do
        insert $(MP1_{cur}, p)$ into $L_{MP\_PAIRS}$
        mark $p$ (in list $MP_2$) as processed
      end for
    end if
    for all $p \in MP_2$ do
      if $p$ is not marked as processed then
        insert $(\emptyset, p)$ into $L_{MP\_PAIRS}$
      end if
    end for
  end while
  return MP_PAIRS: tuples of aligned pairs of masterpeaks.
```

In this approach we simply check for masterpeaks in both groups that have similar m/z values. An obvious problem here is that peaks can be assigned more than once and no similarity measure is used to increase the quality of the assignments.

Approach 2: Bipartite Graph Matching

In the second approach we re-formulate the problem as a *Minimum Weight Maximum Cardinality Bipartite Matching* (assignment) problem. We are given the two sets of masterpeaks (MP_1, MP_2) which can be seen as vertices of a graph. This graph $G = \{V, E\}$ is bipartite because the vertex set V is subdivided into two sets.

The goal is to find the largest possible number of matches between similar elements of the two sets. The similarity is measured by using a cost function that defines the cost for connecting two particular vertices (masterpeaks). The cost is defined as:

$$c(p,q) = \begin{cases} \infty & \text{if } |\mathrm{m/z}(p) - \mathrm{m/z}(q)| > 2 \\ \infty & \text{if } |n(p) = 1 \text{ or } n(q) = 1 \\ |\mathrm{m/z}(p) - \mathrm{m/z}(q)| \cdot s(p,q) & \text{otherwise} \end{cases} \tag{3.7.5}$$

where p and q are masterpeaks, m/z$(\cdot)$ returns the m/z position of a masterpeak, $s(p,q)$ returns the shape similarity of the peaks of p and q and $n(x) = 1$ if masterpeak x is classified as noise. Note that $c(\cdot)$ can be extended to incorporate other masterpeak features.

Bipartite Graph Matching

Let $G = (V,E)$ be a bipartite graph such that there is a partition $V = MP_1 \cup MP_2$ and every edge in E has one endpoint in MP_1 and the other in MP_2. Let $M \subseteq E$ be a matching such that no vertices are incident to more than one edge in M. We also define the cardinality $|M|$ to be the number of edges in M and let $c : E \rightarrow R$ be the cost function on the edge of G (as defined in Equation 3.7.5). The cost (weight) of a matching is the sum of the cost of its edges, that is, $c(M) = \sum_{e \in M} c(e)$

The *Minimum Weight Maximum Cardinality Bipartite Matching* (MWMCB Matching) problem is thus to find the matching on graph G such that $|M|$ and $c(M)$ are minimized where $c(M)$ is the MWMCB weight ((Papadimitriou and Steiglitz, 1982; Mehlhorn and Naher, 1999)).

Bipartite Graph Matching - Solver

Now this problem can be solved using the *Hungarian Algorithm* (Kuhn, 1955; Munkres, 1957) or by expressing it as an *Integer Program* and solving it by the inner point method as follows:

Min $\sum_{i,j} c_{ij} x_{ij}$

subject to:

1) $\sum_j x_{ij} = 1$, $i \in A$

2) $\sum_i x_{ij} = 1$, $j \in B$

3) $x_{ij} \geq 0$, $i \in A, j \in B$

4) x_{ij} integer, $i \in A, j \in B$

where $c_{i,j}$ is the cost for an edge between vertex i and j and $x_{i,j} = 0$ if an edge exist between vertex i and j and 0 otherwise.

Preprocessing 2: Calculation of Masterpeak Property Differences

Algorithm 6 Calculate Masterpeak Property Differences

```
Require: Given: $L_{MP\_PAIRS}$
  while $L_{MP\_PAIRS}$ has more elements do
    $MPP \leftarrow$ next element $(MP_1, MP_2)$ from $L_{MP\_PAIRS}$, $MP_1 \in G_1$, $MP_2 \in G_2$ $L_{MP\_PAIRS}$, $MP_1 \in G_1$, $MP_2 \in G_2$
    d $\leftarrow D_{JS}(MPP)$
    if $h(MP_1) < h(MP_2)$ then
      $d \leftarrow -d$
    end if
    insert $(MPP, d)$ into $L_{MP\_PAIRS\_W/\_DISTANCES}$
  end while
  sort $L_{MP\_PAIRS\_W/\_DISTANCES}$ by distance d
  return $L_{MP\_PAIRS\_W/\_DISTANCES}$: a sorted list containing tuples of aligned pairs of masterpeaks with their respective distances.
```

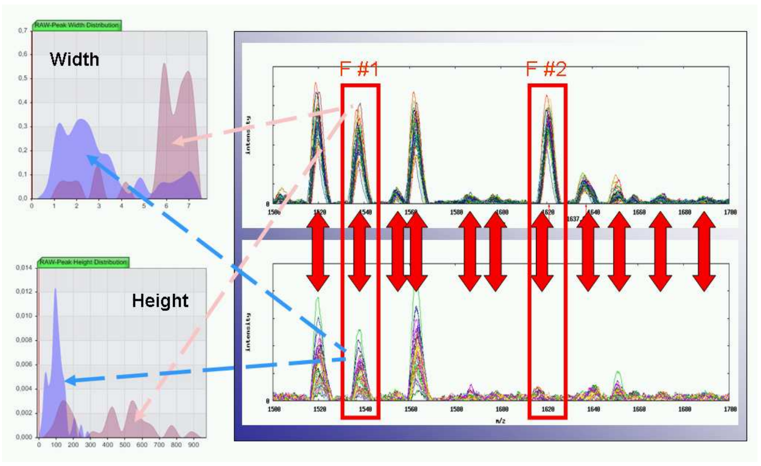


Figure 3.7.17: Right: Aligned masterpeaks of two groups (top: men, bottom: women). Left: Width (top) and height (bottom) distributions of the two masterpeaks of feature "F#1". Colored in blue are the distributions of the peaks associated to the bottom F#1 masterpeak, the distributions for the peaks of the top F#1 masterpeak are colored in purple.

This step calculates the distance of aligned Masterpeak Pairs (MPPs) with respect to the Jensen-Shannon (JS) divergence. For each masterpeak property (e.g. heights of involved raw peaks) of the MPP the distribution is calculated (in the discrete case this would be a histogram). Then the difference of the two distributions is stored in a list ($L_{MP\_PAIRS\_W/\_DISTANCES}$). (See Algorithm 4.) Figure 3.7.17 gives an example: the red arrows indicate the Masterpeak assignments. The distributions for heights and widths (of the masterpeak's raw peaks) of a MPP are shown on the left. (For a similar approach see (Tibshirani et al., 2004).)

3.7.2 Other Approaches

Other alignment approaches have been described in section 3.6.2.

3.7.3 Computational Complexity Analysis of Alignment and Feature Detection

The naive implementation has a complexity of $O(n^2)$ compared to the Hungarian Algorithm that runs in $O(n^3)$), n being the total number of masterpeaks (in both groups). However, the former approach can produce alignments that align a masterpeak of one group to more than one masterpeak of the other group. The Hungarian approach only produces correct assignments and optimizes for the best assignments possible.

Since we typically have some hundred masterpeaks the number of assignments between them is usually relatively small (about 100). Therefore, the cubic runtime is feasible in this application.

The second step, calculation of distances runs in $O(n)$.

3.8 Extracting Fingerprints

3.8.1 What are Fingerprints ?

A Fingerprint, with respect to this thesis, is a set of masterpeaks of a particular set of spectra (for example spectra of lung cancer patients of a certain age) that are not present (or differentiate) in some control group (for example healthy patients of the same age). That is, given an unknown spectrum S (of a patient P) and a fingerprint F of, say a particular disease, we can judge just by checking the occurrence of the peaks of F whether the patient P is likely to have this disease. The following sections will explain how we determine a fingerprint given two sets of spectra (e.g. cancer patients vs. healthy patients).

3.8.2 Our Approach

After the preprocessing steps we now have information about assigned masterpeaks from the two patient (spectra) groups. With these information we can analyze for patterns (fingerprints) by detection and subsequent selection of significant features.

1. *Creation of Fingerprint*
    - Requires *Feature Detection*, described in the previous section and detects potential features that can be used to discriminate two groups based on their properties (e.g. differences in average height, see Fig. 3.6.13)
    - *Feature Selection:* Selection of an optimal subset of features detected (see Figure 3.18(a)).
2. *Reduce Complexity:* Dimensionality Reduction of fingerprint data by projecting fingerprint data to a low-dimensional space (see Fig. 3.18(b)). This is done because it is usually not reliable to cluster in high dimensions.
3. *Evaluation by clustering:* Clustering of low-dimensional projections to get a performance measure. The clusters found can then be used to derive classification rules. (See also section 4.3.3.)

The feature detection step identifies a set of masterpeaks that differ significantly in particular properties (e.g. height, width) between two groups of spectra.

Feature Selection

Generally speaking, feature selection approaches try to find a subset of the original features of the given data. Thus, this step selects a small and efficient[16] set of features from the previous feature detection step. This step is also called modeling since we build a reduced model of the (real) full feature set. By removing irrelevant and redundant features from the data, feature selection helps to improve performance of learning models by

- Reducing the effect of the *curse of dimensionality*

[16]The less features used while maintaining the same discrimination power the more efficient is the set.

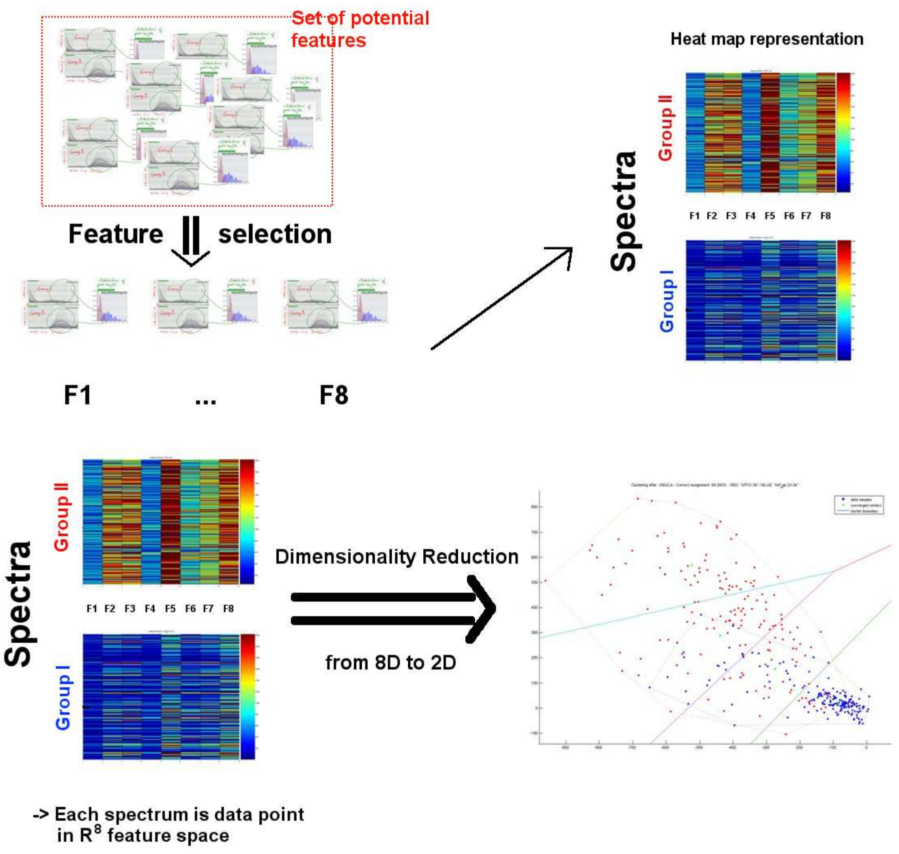


Figure 3.8.18: This shows the feature selection (FS, left) and dimensionality reduction (DR, right) process. First, in the FS stage, from a set of potential features (pairs of masterpeaks, top image) the best x are selected ($x = 8$ in this example). Thus, each spectrum is projected onto a point in $\mathbb{R}^x$. These points are then projected during the DR stage into $\mathbb{R}^2$

where most clustering algorithms are easily applicable.

- Enhancing generalization
- Speeding up the learning process
- Enabling the interpretation of the model

The core idea is to apply a mapping of the high-dimensional data space into a space of fewer dimensions. Two widespread strategies are filtering (e.g. information gain) and wrapping (e.g. genetic algorithm) approaches. In classification applications, feature selection may be viewed as a discrimination problem where one aims to emphasize the class separability by a judicious choice of feature parameters. Ideally, the extracted features should reveal some unique non-redundant characteristics that are most effective in discriminating between classes.

The selection of features is necessary because (a) it is mostly computationally infeasible to use all available features in the subsequent machine learning algorithms and (b) problems emerge when limited data samples but a large number of features are present (this relates to the so called *curse of dimensionality*). It can be shown that optimal feature selection requires an exhaustive

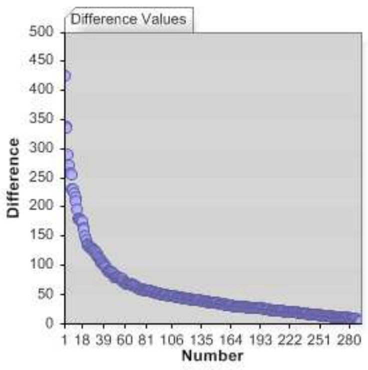


Figure 3.8.19: This figure shows the histogram of the JS-differences of the height distributions of masterpeaks of a particular masterpeak assignment analysis. Experiments have shown that this histogram can be best approximated by an *Extreme Value Distribution* (also known as Fisher-Tippett Distr., (Fisher and Tippett, 1928)).

search of all possible subsets of features, which becomes infeasible if large numbers of features are available. This means that usually a satisfactory set (as opposed to the optimal set) of features is sought for.

Feature selection is a technique used in the machine learning community since years for building robust learning models. We have not tried to come up with a new better feature selection algorithm. Instead we compared many of the most popular existing algorithms and evaluated their performance when applying them to our feature data. The next section shows these evaluation results

3.8.3 Evaluation of Feature Selection Algorithms

This section shows the results of the feature selection algorithms. We created two groups ($n = 150$ spectra each) consisting of females (a) taking oral contraceptives and (b) not taking oral contraceptives. We pre-processed all spectra and performed peak picking and Masterpeak analysis as described. Out of these results we created an initial fingerprint of the best discriminating peaks that have a JS difference (see section 3.7) above 100 (n=34) (see Fig. 3.8.19).

In the remaining, algorithms labeled with the prefix *WEKA* are used from the WEKA 3 Toolkit ((Witten and Frank, 2005)). If not otherwise stated we used the standard parameters proposed by the WEKA team.

Note that this dataset is easy to classify since the features extracted are already quite powerful thanks to our pipeline. This can be verified by looking at the surprisingly good performance of fingerprint A2 (correct classification of 80%, see Table 3.8.2, consisting only of randomly chosen features (out of the Top 100 features).

Starting from the initial 34 feature fingerprint we used the following algorithms to perform a feature selection (For a more detailed description see (Witten and Frank, 2005)):
3^{rd} party feature selection algorithms used in this thesis:

- *CfsSubsetEval*: "Attribute Evaluator": Evaluates the performance of an attribute (sub-)set by predictive ability of each feature while penalizing correlation between them. This algorithm needs a "Search Method" for the subset selection.

FID	*FeatSel Algo*	*#Feat.*	*Performance*[a]
A1	Initial Set	34	89.3 %
A2	Random Selection[b,c]	12	79.3 %
A3	WEKA::CfsSubsetEval::BestFirst[c]	12	89.7 %
A4	WEKA::CfsSubsetEval::GA[c]	12	90.8 %
A5	WEKA::ChiSquaredAttribEval::Ranker	12	90.0 %
A6	WEKA::InfoGainAttributeEval::Ranker	12	90.0 %
A7	WEKA::SVMAttributeEval::Ranker	12	92.7 %
A8	Merged[d]	6	90.2 %

Table 3.8.2: Results of selection experiments. The performance is evaluated by a WEKA::ADTree classificator. [a]: Performance: correct classifications. [b]: Twelve features were selected randomly. [c]: This was repeated 100 times and the average value of the classification result was used. [d]: The features occurring in each fingerprint A3..A7 were used in this fingerprint.

- *BestFirst*: "Search Method": Greedy Hill Climbing.
- *GA*: "Search Method": Genetic Algorithm as described in (Goldberg, 1989).
- *CfsChiSquaredEval*: "Attribute Evaluator": The attribute performance is the chi-squared statistic with respect to the class.
- *Ranker*: "Search Method": Ranks attributes by their individual performance values computed by the "Evaluator Methods".
- *CfsInfoGainAttributeEval*: "Attribute Evaluator":
  InfoGain(Class,Attribute) =
  H(Class) - H(Class — Attribute).
- *SVMAttributeEval*: "Attribute Evaluator": Computes the attribute performance by using an SVM classifier (see (Guyon et al., 2002).)

The performance was measured by the WEKA ADTree classification algorithm (Freund and Mason, 1999), which gives a deterministic, traceable and comparable result[17]. The fingerprint A8 was compiled by taking the features that occurred in all fingerprints A3 ... A7 thus representing the intersection.

Table 3.8.2 shows the performance of the individual algorithms. For the reduced fingerprints we used the Top 12 features ranked by the respective algorithms, since these seem to be the number of the dominant features with respect to the scores (data not shown).

What is striking here is that every fingerprint except the randomly chosen features (A2) performs almost identically well. This is not surprising since all of them share six core features (fingerprint A8), which even by their own reach a competitively high score. These six features are necessary since using fewer features results in scores below 85% (data not shown). We therefore conclude that (a) all of the tested feature selection methods are capable to produce smaller sets of features while maintaining their performance and (b) even smaller feature (sub-)sets can be built by using them in a *voting schema.*

Dimensionality Reduction

The fundamental assumption for Dimension Reduction (DR) is that the informative part of the data (at least approximately) lies on a linear or nonlinear

[17]This means, the resulting trees can be understood and analyzed by humans, opposed to matrix transformations or resulting Neural Networks.

manifold $\mathcal{M}$ of smaller dimension than the original d-dimensional data space $\mathcal{X}$. Its primary goal is to find a representation of $\mathcal{M}$, allowing to project the data $(X_n)_{n=1}^N$ on it and thus obtain a low-dimensional, compact representation of the noisy sample. The challenge is to find a linear or nonlinear mapping $f : \mathbb{R}^d \to \mathbb{R}^m$ with $m \leq d$ such that reconstruction error of the sample,

$$\mathbb{E}_\lambda[\{X_n\}_{n=1}^N] = \frac{1}{N}\sum_{n=1}^{N} \lambda(X_n, X_n')$$

is acceptably small. Here, $X'_n = f(X_n)$ is the reconstructed vector for X_n and λ a suitable distance measure in $\mathcal{X}$. At best one will get rid of some computational obstacles:

i) efficiency and classification performance

ii) measurement, storage and computation costs

iii) ease of interpretation and modeling

The problem of DR technically decomposes into two tasks: First one has to determine elements from the target space. Second, one has to construct a basis of the target space from these elements. Hence DR is often achieved by semi-parametric feature extraction.

In this work we use DR techniques to reduce the dimensionality of the fingerprint feature vectors by projecting from the high dimensional feature space to some low-dimensional embedding. This enables subsequent clustering algorithms to work reliably in two or three dimensions, since they usually are not designed to cluster points in high-dimensional space (see e.g. the *curse-of-dimensionality* and section 4.3.3).

As in the case of *feature selection* for example in the machine learning community much effort has been put into the development of sophisticated (general) DR algorithms. Therefore, we did not develop an own approach but evaluated existing ones with the fingerprints gained from the feature selection step. We can show that it is possible to reduce the complexity of these fingerprints even more by only losing very little classification power. This results in a classifier being a line or a plane and gains comprehensible clustering possibilities in R^2 or R^3 which are easily visualizeable.

All of the dimensionality reduction (DR) algorithms project the n-dimensional fingerprint data[18] to $d = 2$ dimensions[19]. Subsequently the resulting points are clustered by k-means clustering[20] and a *goodness of clusters* (GOC) score is calculated as follows:

$$GOC = \frac{\sum_{i=1}^{n} \frac{|C_i|}{max(|C_{i,G1}|,|C_{i,G2}|)}}{n} \tag{3.8.6}$$

where n: number of clusters, $|C_i|$: number of total points in cluster C_i, $|C_{i,Gx}|$: number of points of group x in cluster i.

For the evaluation we used some widely used algorithms of three classes:

[18] n being the number of features

[19] The performance can be increased by projecting into $d = 3$ dimensions (data not shown) but for easier visualization we took only two dimensions.

[20] $k = 5$

DimRed Algo	*A1*	*A2*	*A3*	*A4*	*A5*	*A6*	*A7*	*A8*
PCA	0.82	0.63	0.81	0.82	**0.83**	0.82	0.78	**0.83**
MDS	0.85	0.63	**0.87**	0.86	0.82	0.85	0.83	**0.87**
ISOMAP	0.82	0.66	0.83	**0.84**	0.78	0.77	0.83	0.83

Table 3.8.3: Results of Dimensionality Reduction experiments. Shown are the GOC scores (see section 3.8.3). Best GOC score for a dimensionality reduction algorithm is shown in bold. A1..A8 refers to the FIDs from Tab. 3.8.2.

- *Principal Component Analysis (PCA)*: An orthogonal linear transformation that projects data to a lower dimensional space such that the basis of the new coordinate system is spanned by the dimensions of greatest variance.
- *(non-metric) Multidimensional Scaling (MDS)*: MDS non-linearly projects high-dimensional data points to a low dimensional space while (as good as possible) reproducing the distances between the original data points. (See (Shepard, 1962) for further details.).
- *Isometric mapping of data manifolds (ISOMAP)*: Graph-based version of MDS that tries to maintain the geodesic distances (opposed to Euclidean distances in original MDS). (See (J.B. Tenenbaum and Langford, 2000) for more details.)

Of course, other (more recent and/or complicated) algorithms could have been used as well but in this thesis we focus on methods that are well studied and their properties are well understood. This is due to the topic of determining diseases (such as cancer) from humans: we rather have some good and robust method that we can fully analyze and understand than a black box that we cannot analyze anymore.

Table 3.8.3 shows the results of the procedure described above: the *Goodness of Clusters* score (see Equation 3.8.6) is on average above 0.80 while the non-linear methods perform superior over the linear PCA dimensionality reduction. It is remarkable that these projections to two dimensions allow subsequent cluster algorithms to find that much structure in the data. We therefore think that this approach is very well suited to build reliable, robust and traceable classifiers.

3.8.4 Other Approaches to Pattern Detection

Pattern detection algorithms have become widely used in almost all disciplines that somehow deal with mining of data. Reviewing all these different approaches is definitively outside the scope of this thesis, though we have given some remarks about other algorithms above. To get some other insights into the area we refer to a beautiful review of “36 years on the pattern recognition front” by (Pavlidis, 2003) and an exhaustive article by (Yang and Honavar, 1998) about feature selection.

3.8.5 Using the Fingerprints

The fingerprints (patterns) created in the previous steps now enable two things:

- First, they can be used to classify unknown spectra, e.g. if a fingerprint for a particular cancer has been found, a new spectrum can be checked whether it contains peaks similar to those the fingerprint consists of (see chapter 4).

- Second, the components (peaks) of a fingerprint can be analyzed further, that is, the underlying molecules can be determined and advanced studies can be made to investigate the role of these molecules in the human body (see section 6.2.3).
- Third, they can be used to cluster spectra to detect sub-groups even within a group, e.g. a particular type of cancer (see section 4.3.3.

3.9 Complexity Analysis

Using the single complexity analyses from the previous chapters the full complexity analysis results in:

Preprocessing: $O(n \log n)$, n being the number of sample points.

Peak Picking: $O(n)$, n being the number of sample points.

Peak Registration: $O(m \log m)$, m being the number of peaks.

Feature Extraction: $O(p^3)$, p being the number of masterpeaks.

From empirical studies we know the approximate value ranges of these variables:

n**:** 10^4

m**:** $\frac{n}{10} = 10^3$

p**:** $\frac{m}{10} = \frac{n}{100} = 10^2$

Chapter 4

(Bio-)Medical Applications

Contents

This chapter describes the range of applications the algorithms introduced in the previous chapter can be applied in and pinpoint some common pitfalls when interpreting the results. Further, we described the dataset we used throughout this thesis and for the development and benchmarking of our methods.

4.1 Data Used

This section describes the biological data we are using in this thesis to perform our experiments and check our algorithms. Since the actual data we are using is derived from biological samples (blood) we also give some background to what must be taken care of when using such material.

4.1.1 Some Remarks on Blood

During the previous years in proteomics driven cancer research much emphasis has been given to blood analysis. Usually, plasma or serum data from cancer patients was compared against samples from matching healthy subjects to detect differences in the proteome. Many interesting results have been obtained by the evaluation of MS profiles, as the following examples show:

- (Zhang et al., 2004) identified and validated three early-stage ovarian cancer biomarkers through MS analysis of the serum proteome. The combination of the three identified biomarkers (apolipoprotein A1, atrucated form of transthyretin and a cleavage fragment of inter-alpha-trypsin inhibitor heavy chain H4) is superior to CA-125 (cancer antigen 125, a mucinous glycoprotein) alone in terms of sensitivity. CA-125 is a standard tumor biomarker for ovarian cancer.

- (Honda et al., 2005) conducted a MS-based analysis of 245 plasma samples from pancreatic cancer patients and controls. They found a fingerprint (peak pattern) consisting of four peaks that was sufficient to discriminate cancer patients from healthy subjects with a sensitivity of 91%. Combining this fingerprint with CA-19-9 (carbohydrate antigen 19-9) this sensitivity increased to 100%.
- (Hong et al., 2005) developed a MS-based approach for determining the severity of multiple myeloma from serum of 64 newly diagnosed multiple myeloma patients.

In addition to these reports excellent reviews are available, e.g. (Rosenblatt et al., 2004; Thadikkaran et al., 2005; Omenn, 2006; Ebert et al., 2006).

The advantage of blood is that it has a very high protein concentration. The bad news is, however, that only 22 proteins account for about 99% of its protein content. These belong to the so-called high-abundance protein species, including albumin (approximately 60% of the total protein in normal human plasma), transferrins, immunoglobulins, etc. When interested in low-abundance circulatory proteins, for example produced by tumors and especially early-stage tumors, one must keep in mind that these will account for less than 1% of the blood proteins. So, how to enrich the low-abundance proteins become extraordinarily important in blood-based cancer proteomics investigations.

As mentioned above, blood can be analyzed by two methods: (a) using the plasma which is the liquid portion of blood in which the cells are suspended or (b) using serum which is the fluid that remains after clotting proteins are removed from plasma. Although, the advantage of using plasma is that it contains more proteins and its protease activity is inhibited the same fact is also a disadvantage: it is very difficult to detect low-abundance proteins in the presence of many other (common, e.g. clotting) abundant proteins.

In principle, mass spectrometry-based proteomics analysis of blood can be performed on both types, plasma and serum. However, there is still no agreement in the community which is best:

- Regarding the ongoing enzymatic activity in serum (Koomen et al., 2005) indicated that serum is not well suited for proteomics experiments because even proteins that are not involved with the biologically relevant pathways are cleaved as well by non specific proteases.
- (Tammen et al., 2005) recommended the use of platelet-depleted EDTA or to citrate plasma for the analysis of the low-molecular-weight proteins.
- It was shown in (Villanueva, Shaffer, Philip, Chaparro, Erdjument-Bromage, Olshen, Fleisher, Lilja, Brogi, Boyd, Sanchez-Carbayo, Holland, Cordon-Cardo, Scher and Tempst, 2006) that - when focused on analysis of low-molecular-weight proteome - serum is superior to plasma as a source of diagnostic information in terms of peptidomics.

In this study, blood *serum* served as a basis for the acquisition of data.

4.1.2 Serum Data Used

The data was acquired by our partner organization the *Institute of Laboratory Medicine, Clinical Chemistry and Molecular Diagnostics at the University Hospital Leipzig.* The biological foundation consists of frozen serum samples of 770 apparently healthy blood donors who served as basis for the mass spectrometric investigation. The study group has been described elsewhere (Kratzsch et al., 2005).

Human serum is particularly susceptible to any confounding factor, for example, (Boguski and McIntosh, 2003) showed that it does make a difference whether the subject is sitting or recumbent during blood taking and can induce a change by 10% in total protein concentrations. Further, the proteome profiles can also be significantly influenced by the pre-analytical process such as collection of samples (time-frames) and how the processing is done, e.g. variables of clotting tube types, freeze-thaw cycles, centrifugation, storage time and temperature, and so forth (Banks et al., 2005). Based on these results, (Baumann et al., 2005) developed a protocol for preanalytics that was used in this study.

Proteome fractionation: After thawing on ice peptide and protein purification and fractionation was performed using a magnetic bead-based separation technique (ClinProt™, Bruker Daltonics, Leipzig, Germany) with specific surface functionalities (MB-IMAC Cu, MB-WCX, MB-HIC C8).

Chemicals & Consumables: Gradient grade acetonitrile, ethanol, acetone were obtained from J.T. Baker (Phillipsburg, USA); trifluoroacetic acid was purchased from Sigma-Aldrich (Steinheim, Germany). Peptide and protein calibration standards, a-cyano-4-hydroxycinnamic acid were purchased from Bruker Daltonics (Leipzig, Germany). Peptide preparations were done in 0.2ml polypropylene tubes (8-tube strips) from Biozym (Hess. Oldendorf, Germany). The MALDI-TOF AnchorChip™ target (four spots) was purchased from Bruker Daltonics (Leipzig, Germany). The protein calibration standard mix (Part No.: 206355 & 206196) used for the spiking experiments was bought from Bruker Daltronics (Leipzig, Germany).

Hardware Configuration: ClinProtTM Robot & Autoflex Linear MALDI-TOF Mass Spectrometer (Bruker Daltonics, Germany)

Mass Spectrometry: Mass spectra were recorded by the flexControl™ 2.0 Software (Bruker Daltonics, Germany). The settings were applied as follows: Ion source 1: 20 kV; ion source 2, 18.50 kV; lens, 9.00 kV; pulsed ion extraction, 120 ns; nitrogen-pressure, 2500 mbar. Ionization was achieved by a nitrogen laser (λ=337 nm) operating at 50 Hz. For matrix suppression a high gating factor with signal suppression up to 500 Da was used. Mass spectra were detected in linear positive mode. Spectral data were combined with beforehand surveyed epidemiological and clinical metadata in a Microsoft SQL Server™ database to provide highly differentiated classification criteria (such as age, gender, or even peptide admixture).

Overall, about 8500 spectra were acquired during data collection. To cope with the amounts of raw data and data generated during the analyses, and to enable access to one structured storage (as opposed to many different files), a Microsoft SQL Server™ database was used.

4.2 Statistical Remarks

4.2.1 Measurement Fundamentals

Campbell defines measurement as the “assignment of numerals to represent properties” (Campbell, 1920). This very general statement fits with the common perception of this term and is used in many different application domains: people are measuring the diameter of the sun, the mass of an electron, the intelligence of a human being, or the popularity of a television show. In other words, measurement is the objective representation of objects, processes, and phenomenon (Finkelstein and Leaning, 1984). Measurement captures information about a system through its properties (also called characteristics, features, or attributes) which can be either directly or indirectly observable. Additionally, there exist relationships between the properties and the elements (i.e. measured representations) of a system. Thus, a system X is defined by the properties x_i chosen to represent it:

$$X = (x_1, x_2, \dots, x_i)$$

Mostly, there are three elements present: a property to be determined (P), a measured quantity (M) and a relationship between these two quantities:

$$M = f(P)$$

Figure 4.2.1 is a graphical representation of the relationship associated with a measuring process.

Since this formulation only addressed the properties selected to represent the system one easily notices that, although being objective, it is an abstraction, and many important properties of the systems might not be included. Property selection is crucial since the validity of a system measurement is influenced by the number of properties used in the measurement. Clearly, properties affect the validity of a measure. Therefore, formalized frameworks and theories are required to clarify concepts about measurement within a particular domain.

Obviously there is a catch here with regard to property selection: In order to measure a system properly, one needs to know something about it - however, the main reason to measure a system is to gain an understanding of it. Therefore, for most complex systems the properties that best define such a system are unknown, inaccessible, or only visible as an outcome. Measurement of these systems requires use of a proxy or indirect measuring method which is essentially a model or approximation of the system property of interest. The process of deriving these proxies usually involves reducing complex aspects of a system into understandable, measurable components.

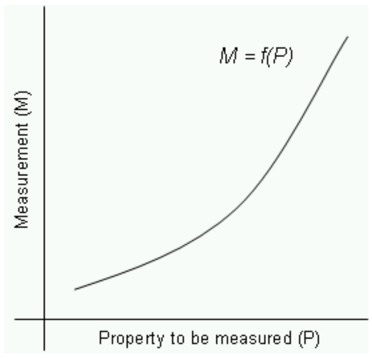


Figure 4.2.1: A monotonic relationship associated with a measuring process.

Having defined or selected the properties the subsequent measurement can be thought of as a process of assigning numbers (or generally symbols) to the properties of a system such that these symbols reflect the underlying relationships (or nature) of the properties (Caws, 1959).

Having performed the measurements one should analyze its main characteristics (Geisler, 2000), namely:

1. Validity: characterizes how well a measure reflects the system properties it was supposed to represent.

2. Reliability (or precision): addresses the consistency or repeatability of the measurement process.

3. Amplitude: how well a measure represents abstract or higher order constructs and complex properties.

4.2.2 Making Errors

Measurements are generally made through the use of a measurement instrument, which is based on a scale that should have the same underlying relationships as the system property being measured. Formally put, a scale is a predefined mapping from one domain to another (e.g. the volume expansion of mercury to degrees of Celsius). The mappings can of course induce uncertainty e.g. through the use of fuzzy scales to represent the degree to which a property is considered present (Benoit and Foulloy, 2003). Obviously, the construction of a scale can be a source of error as well as each observation itself, which is a random variable with an underlying distribution (Potter, 2000). Basically, there are four primary sources of measurement error:

- Random error: non-deterministic variation from any source impacting the system including the system itself.

- Systemic error: derives from construction of the measure or definition of the measurement process and comes in form of measurement bias.

- Observational error: the oversight of key system properties requiring measurement or using the wrong measures for identified system properties.

- Experimental error: the influence of environment conditions (such as temperature, air pressure, or response time of the operator) which affects the measurement process

These errors create divergences between the perceived state of a system and the true state and can yield misleading insights and thus, must be addressed in any measurement framework. They will be part of the measurement process even when the system is welldefined (Krantz et al., 1971). Error is an inescapable feature of measurement (Mitchell, 2003) and can be partially addressed with statistical theory as described in chapter 3.

4.2.3 Experiment and Inference

As mentioned in the previous sections when experiments are carried out, the results of the measurements and the subsequent statistical analyses are often stated in the language of mathematics or, more precisely, in that of the theory of probability. These mathematical statements have the beauty of being objective, precise and clear. On the other hand, this might induce people to hide inadequate experimentation behind a brilliant (mathematical) facade. A fact, unfortunately often seen in scientific publications. Let discuss a simple example: data consisting of two columns of numbers, say x and y, can always be subjected to calculations known as linear regression. This works as a foundation for correlation analysis and to various other tests of statistical significance. Despite their mathematical preciseness and undoubtedly correctness, inferences drawn from these results may be incorrect, thoroughly misleading, or failing to call attention to the basic insufficiency of the experiment. This usually has two main reasons:

- assumptions underlying the statistical procedure are not fulfilled
- problems connected to the data were of a completely different type from those for which the particular statistical methods provide useful information.

Indeed, most data sets provide some useful information, but this is no guarantee that the information actually desired has been obtained.

In most cases the goal of statistical analyses is to draw inferences from the particular to the general and often people are not familiar with the problems of inductive inference, which is closely tied to this. R. A. Fisher has pointed to a basic and most important difference between the results of induction and deduction (Fisher, 1959) which we recall briefly and illustrate by two small examples. By using deduction, conclusion based on correct partial information are always correct, despite the incompleteness of the premises. Let us use a well-known theorem from geometry as an example: the sum of the angles of a plane triangle always equals to 180 degrees. This does not necessitate information as to whether it is isosceles or not. If any information of this type is subsequently added, it cannot possibly alter the fact expressed by the theorem.

As an counterexample, inferences drawn by induction from incomplete information may be entirely wrong, even when the information given is unquestionably correct. Let us use a simple example from physics. Suppose we were given the data of Table 4.2.3 on the pressure and volume of a fixed mass of gas. Analyzing the data one might infer (by induction) that the pressure

Molar volume (liters)	Pressure (atmospheres)
0.182	54.5
0.201	60.0
0.216	64.5
0.232	68.5
0.243	72.5

Table 4.2.1: Volume-Pressure Relation for a gas, an apparently proportional relationship

of a gas is proportional to its volume (of course a completely erroneous statement). What went wrong ? The answer is simply that another important item of information was omitted, namely that each pair of measurements was obtained at a different temperature, as indicated in Table 4.2.3. Of course, this example is artificially constructed and extreme but it emphasizes the basic problem in inductive reasoning: the data not only has to be correct but also complete to enable correct inference. In this simple example the missing piece of information was easily identifiable because we have a good understanding of the physical background. What researchers need to be aware of is that

Molar volume (liters)	Pressure (atmospheres)	Temperature (degrees Celsius)
0.182	54.5	15.5
0.201	60.0	25.0
0.216	64.5	37.7
0.232	68.5	50.0
0.243	72.5	60.0

Table 4.2.2: Volume-Pressure-Temperature Relation for a gas

they normally are not familiar with the physical relations of a system they are analyzing and therefore recognize the danger of drawing false inferences from incomplete, though correct information.

4.2.4 Statistical Significance Testing

Statistical significance testing has been conducted by scientists (at least) since the early 1700 (McLean and Ernest, 1998). However, the concept does not seem to be entirely clear to all scientists and consequently it is often misused or its results misinterpreted[1,2] (Thompson, 1994). As early as 1931, (Tyler, 1931) noted the misuse of statistical significance:

> "[...] we are prone to conceive of statistical significance as equivalent to social significance. These two terms are essentially different and ought not to be confused."

A statistical significance test basically determines whether or not a difference exists between variables and was advanced by Fisher, Neyman and Pearson (see (Fisher, 1922; Neyman and Pearson, 1928) and (Lehmann, 1993) for a discussion). For example, a researcher who believes that drug A is more effective than drug B formulates a so-called *null hypothesis* that the two drugs are equally effective. He would then attempt to disprove this claim by conducting a study with two randomly chosen patient groups, each group being treated with one of the drugs. Then he would somehow measure the effectiveness of the respective drugs and compute a probability value, p. This value p reflects the probability of obtaining the results if actually the null hypothesis was true. If p is less than a previously set value α (typically $\alpha = 5\%$), the results are said to be significant, which means that (probably) the two drugs are not equally efficient. On the other hand, if p exceeds α the scientist would report that the study is not significant (thus the null hypothesis holds), meaning that the two drugs may or may not be equally effective.

Opponents of this *rejection framework* claim that this approach is counterintuitive and reporting on what one found instead of what can be rejected would also be absolutely appropriate (see e.g. (Cohen, 1990)). Further, it is quite important to note that the commonly used level of $\alpha = 5\%$ (described

[1]In a study by (Anthony, 1996) the use of statistics in papers from high-quality medical research journals were analyzed. He reports that statistical errors and misunderstandings of statistical concepts are almost the norm rather than the exception. Errors were found in more than 45% of all papers reviewed.

[2]One in 20 studies using a significance level of 95% are likely to detect some statistically relevant finding by chance alone, see e.g. (McCloskey, 1995)

as a *Gold Standard*(McCloskey, 1995)) is arbitrary and to remember that improbable outcomes can, and do, occur by chance (which is why many of us buy lottery tickets). Therefore the level of statistical significance should not be set in the abstract but in the light of current knowledge, since it has important implications for the reporting of results.

Another quite important fact to be aware of when performing statistical testing is the influence of (a) the sampling itself and (b) the size of the sample. Both can greatly affect the outcome of a test: when investigating differences between two groups usually not the whole population can be examined but, instead, a (random) sample is taken from each group. It is important to realize that differences in these samples may be just the result of the sampling (Rudy and Kerr, 1991).

Another misbelife is that increasing the sample size will achieve a higher significance: large samples may also introduce new problems, since even well selected samples are never identical. However, (Bakan, 1970) shows that it is almost certain in large samples to find statistically significant differences in one or more variables - the important question here is whether these are important or just occur by chance. Therefore, the real value of large samples is that they increase confidence in the reliability of the results (see *Confidence Intervals* in section 4.2.5).

To summarize: in clinical research emphasis should be laid on the actual size of a difference found then on the mere statistical significance, since these are usually of greater importance in practice (see *Effect Size* in section 4.2.5). Therefore, a detected statistical association should be treated like a messenger, suggesting that further research might be valuable.

4.2.5 Advanced Result Analysis

This section summarizes alternative concepts to statistical significance testing. These can be used as stand-alone tools or to provide further overall significance to a study. Three approaches will be briefly described below that can support and/or extend classical statistical significance testing. First, a method called *effect size* that measures the actual relevance of a finding should be calculated and interpreted in all analyses. Second, the precision of a result should always be given, by e.g. a *confidence interval.* Third, the *replicability* of results must be empirically investigated, either through actual replication of the study, or by using methods such as the jackknife or the bootstrap.

Effect Size

The so-called effect size measures the magnitude of the relationship between two variables. Unlike significance tests, these indices are independent of sample size. For example, if a p value reports a difference between a parameter of two groups to be statistically significant, the effect size reports the size of this observed difference and helps to determine whether it is of practical concern. As we have described in the previous sections, given a sufficiently large sample size, it is always possible to show that there is some significant difference. Effects size is a tool to know whether this difference observed really matters. To summarize, effect sizes are important to report and interpret for at least two reasons:

- First, these indices can help judging the practical or substantive significance of results, because improbable events are not necessarily important (see e.g. (Shaver, 1985) famous telephone example).
- Second, reporting effect size facilitates the meta-analytic integration of findings across different studies (of the same topic), and thus comparison.

Interpreting Effect Size

Effect size and the actual *value* of an effect can be interpreted in various ways:

Technical: Effect size is equivalent to a *Z-score* of a standard Normal distribution. For example, an effect size of 0.8 means that the average value of a variable, say IQ, in group A is 0.8 standard deviations above the average value of this variable in group B, and hence the average IQ of person from group A exceeds the IQ of 79% of persons of group B.

Comparative: A quite descriptive way to interpret effect sizes is to compare them to the effect sizes of differences that are familiar. For example, (Cohen, 1969) analyzed the effect size of the height difference between several age groups of (American) girls. He describes a 0.2 effect size between the heights of 15 year old and 16 year old girls as “small”. An 0.5 effect size between 14 and 18 year old girls is described as “medium” and “large enough to be visible to the naked eye”. A “large” effect size of 0.8 is “grossly perceptible” and equivalent to the difference between the heights of 13 year old and 18 year old girls.

Contextual: Cohen does acknowledge the danger of using terms like “small”, “medium” and “large” out of context. (Glass et al., 1981) argue that the effectiveness of e.g. a drug treatment can only be interpreted in relation to other drugs that aim to produce the same effect. For example, a 2% variance-accounted-for effect size will not be very impressive to most researchers in some study dealing with the effect of smiling on the amount of tip in some restaurants. However, (Gage, 1978) cites a study investiging the relationship between cigarette smoking and lung cancer that resulted in roughly the same effect size and noted that

> “Sometimes even very weak relationships can be important [...] [On] the basis of such correlations, important health policy has been made and millions of people have changed strong habits.”

However, it needs to be realized that the recognition of a study result always includes personal, own values of the reader: “As in all of statistical inference, subjective judgment cannon be avoided. Neither can be reasonableness!” (Huberty and Morris, 1988).

Effect Size Measures

There are many different effect size measures to choose from. Most commonly effect size is reported by standardized measures of effect or by unstandardized (but intuitive) measures (e.g. the raw difference between group means or the loss of weight a new diet promises). Thus, standardized measures are typically used when the metrics of variables being studied do not have an intuitive meaning to the target audience, or when results from multiple studies of an area are being combined.

To give an example for a standardized measure let us look at Cohen's d (Cohen, 1988) which is the difference between two means divided by the pooled standard deviation for those means:

$$d = \frac{\text{mean}_1 - \text{mean}_2}{\sqrt{(\text{SD}_1^2 + \text{SD}_2^2)/2}}$$

where mean_i and SD_i are the mean and standard deviation for group $i = 1, 2$.

Note that sample size does not play a part in the calculation that d is heavily influenced by the denominator in the equation. This is incorporated in (Hedges and Olkin, 1985)'s $\hat{g}$:

$$\hat{g} = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{(n_1-1)\cdot SD_1^2+(n_2-1)\cdot SD_2^2}{((n_1+n_2)-2)}}} \cdot \left(1 - \frac{3}{4\cdot(n_1+n_2)-9}\right).$$

Note that both values can also be computed from the value of the t test of differences between the two groups (Rosenthal and Rosnow, 1991; Rosnow and Rosenthal, 1996).

Replication & Bias Estimation

> "If science is the business of discovering replicable effects, because statistical significance tests do not evaluate result replicability, then researchers should use and report some strategies that do evaluate the replicability of their results." (Thompson, 1995)

Empirical evidence for result replicability can be gained either *external* or *internal* (Thompson, 1995). While the external replication involves the creation of a new sample measured at a different time and/or different location, internal replicability is based on comparing recombinations of sub-samples of the full sample at hand. An internal replication is performed to estimate the *bias* (also called imprecision or random error) of the analytical method. It helps to understand the precision of some sample statistics, such as median, variance or percentiles. Methods of measurements are almost always subject to some random variation by using subsets of (jackknife) or drawing randomly with replacement from (bootstrapping) the available data (samples). The following paragraphs introduce these techniques.

Jackknife (Tukey, 1958): The jackknife is a simple method for *approximation of the bias and variance of an estimator.* Basically, the jackknife approach partitions out the impact of a particular subset of the data on an estimate derived from the total sample. In other words, jackknife tries to control for a piece of the sample which may be exerting too much influence on your results due to sampling error. This is done by systematically recomputing the desired statistical estimate leaving out one observation at a time from the sample set. The Jackknife is less general than bootstrap (see below) but easier to apply to complex sampling schemes, such as multi-stage sampling with varying sampling weights. The following introduces the calculation of the jackknife estimator:

Let $\hat{\theta} = T(X_1, \ldots, X_n)$ be an estimator for some (unknown) quantity θ (e.g. mean) and

$$bias(\hat{\theta}) = \mathbb{E}(\hat{\theta}) - \theta$$

Let T_{-i} be the statistic with the ith observation removed.

The jackknife bias estimate is defined as

$$b_{jack} = (n-1) \cdot (\bar{\theta} - \hat{\theta})$$

where

$$\bar{\theta} = \frac{1}{n} \cdot \sum_{i=1}^{n} T_{-i}$$

So the bias corrected estimator is

$$\hat{\theta}_{jack} = \hat{\theta} - b_{jack}$$

It can be shown (Wasserman, 2006) that the bias of $\hat{\theta}_{jack} = O(\frac{1}{n^2})$ which is up to one order of magnitude smaller than that of $\hat{\theta}$. In practice, however, correcting for the bias can obviously increase the standard error - this can be checked for example by bootstrapping (see below) using the biased and the corrected estimator.

To calculate the *variance* by the jackknife approach, note that $\hat{\theta}_{jack}$ can also be written as:

$$\hat{\theta}_{jack} = \frac{1}{n} \cdot \sum_{i=1}^{n} \tilde{T}_i \tag{4.2.1}$$

where

$$\tilde{T}_i = n \cdot \hat{\theta} - (n-1) \cdot T_{-i}$$

The $\tilde{T}$ are called *pseudo values* and are supposed to act as if they were n independent variables. Using equation 4.2.1 the jackknife estimate of $\mathbb{V}(T_n)$ is:

$$v_{jack} = \frac{\tilde{s}^2}{n}$$

where

$$\tilde{s}^2 = \frac{\sum_{i=1}^{n} (\tilde{T}_i - \frac{1}{n} \cdot \sum_{i=1}^{n} \tilde{T}_i)^2}{n-1}$$

is the sample variance of the pseudo-values. Again, it can be shown that v_{jack} is a consistent estimator of $\mathbb{V}(T_n)$ if T is a smooth function of the sample mean - thus the estimate for the median is usually inconsistent. It is important to note that bias correction should only be used if the bias is *much* higher than variance of a statistic.

Bootstrap (Efron, 1979): The bootstrap is a method for *estimating the variance and the distribution of a statistic* $T_n = g(X1, \ldots, X_n)$ (note that T_n needs to be *Hadamard Differentiable*, see e.g. (Shao and Tu, 1995)). In principle it can also be used to estimate some parameter θ. This method first creates an infinitely large *mega* data set by copying the original data set many time. Then a large number of different samples are drawn from this mega set and analyses are performed separately for each sample and the results averaged. Thus, a lot of configurations (including configurations in which an item may be represented several times or not at all) are considered and conclusion about generalization of the results can be drawn. It is a robust alternative to inference based on parametric assumptions when those assumptions are in doubt, or where parametric inference is impossible. Opposed to jackknife the bootstrap gives slightly different results when repeated on the same data.

In the real world we would sample n data points $(X_1 \ldots, X_n)$ from some CDF F and calculate a statistic $T_n = g(X_1 \ldots, X_n)$. Transferred to the bootstrap world, we sample n data points $(X_1^* \ldots, X_n^*)$ from $\hat{F}_n$ and estimate a statistic $T_n^* = g(X_1^* \ldots, X_n^*)$. Drawing n points at random from $\hat{F}_n$ is the same as drawing a sample of size n with replacement from $(X_1 \ldots, X_n)$ (the original data). By the law of large numbers we know that $v_{boot} \xrightarrow{a.s.} \mathbb{V}_{\hat{F}_n}(T_n)$ as $B \to \infty$. It follows

$$\mathbb{V}_F(T_n) \overbrace{\approx}^{O(1/\sqrt{n})} \mathbb{V}_{\hat{F}_n}(T_n) \overbrace{\approx}^{O(1/\sqrt{B})} v_{boot}$$

For the parameter estimation, the number of the bootstrap samples B is usually chosen to be around 200. The algorithm for estimating the variance of some statistic T_n is as follows:

- Given data: $X = (X_1, \ldots, X_n)$
- Repeat the following two steps $i = 1 \ldots B$ times
  1. Draw $X^* = (X_1^*, \ldots, X_n^*)$ with replacement from X
  2. Calculate $T_{n,i}^* = g(X_1^*, \ldots X_n^*)$
- This results in B estimators $(T_{n,1}^*, \ldots, T_{n,B}^*)$ and can be used for various purposes (for variance estimation, for interval estimation, hypothesis testing and so on).

For example the variance estimator is computed by:

$$v_{boot} = \frac{1}{B} \cdot \sum_{b=1}^{B} \left(T_{n,b}^* - \frac{1}{B} \cdot \sum_{i=1}^{B} T_{n,i}^* \right)^2$$

and the estimator for the standard error by:

$$\hat{se}_{boot} = \sqrt{v_{boot}}$$

Jackknife vs. Bootstrap

Since the jackknife only needs n computations it is usually easier computable compared to about 200-300 replications needed for the bootstrap. However, only using the n jackknife samples, the jackknife uses only limited information about the statistic $\hat{\theta}$. It can be shown that asymptotically the estimators of the bootstrap and the jackknife algorithms are in fact equal (see e.g. (Efron and Tibshirani, 1994; Fan and Wang, 1995)). Since the bootstrap method can also be used with small sample sizes (opposed to the jackknife) this method should be favored when only one technique can be applied.

Confidence Intervals

As described above the p value can easily be misinterpreted because it combines information about effect size with information about the precision of the effect size estimate. Opposed to that, confidence intervals offer the estimate of some meaningful parameter (e.g. then mean) and the precision of that estimate.

For example, rather than reporting usage of drug A yields an improvement on a significance level $\alpha < 0.01$, using confidence intervals allows to report that this drug yields an improvement of 20% with a 95% confidence interval of $15 \ldots 24\%$. This means, that the best (point) estimate for this parameter is 20% which equals the observed parameter in this study. The interval endpoints (15% and 24%) reflect the variability of the parameter in this population and are consistent with the observed data: in 95% of replications of the process of obtaining the data the interval will include the parameter. The chosen level of confidence is again arbitrary - common values are $80\%, 90\%$ or 95%.

It is important to realize that the interval endpoints themselves are random variables also estimated using sample data. That is, the confidence interval does not indicate that, given the endpoints, the chance are $X\%$ that the interval will include the parameter (note that it is the confidence interval that is random, not the unknown parameter). However, confidence intervals do have a very appealing feature: even if all the research in an area of inquiry was based on radically erroneous estimates of parameters, the parameters would still emerge across studies as a series of overlapping confidence intervals converging on the same parameter.

Another quite important fact is that taking the intersection of two confidence intervals C_1, C_2 (with level α_1 and α_2, respectively) decreases the power of the new confidence interval to $1 - \alpha_1 - \alpha_2$. This is because the probability that $C_1 \cap C_2$ does not contain θ is the probability that either interval does not contain θ. This is less than or equal to the sum of the probabilities of those two events - $\alpha_1 + \alpha_2$. Therefore, $C_1 \cap C_2$ is a level $1 - \alpha_1 - \alpha_2$ confidence region for θ. Thus, a smaller region is attained, but at a reduced confidence level.

Calculating confidence intervals (CI) for some quantity θ can be done in numerous ways. We will introduce the two main bootstrap-based approaches, that account for the distribution of T_n, namely if T_n is (close to) a normal distribution or if it is not:

(1) If T_n is close to a normal distribution the computation of CI is quite simple:

$$C_n = (T_n - z_{\alpha/2} \cdot \hat{se}_{boot}, T_n + z_{\alpha/2} \cdot \hat{se}_{boot})$$

where

$$z_\alpha = \phi^{-1} \cdot (1 - \alpha)$$

and ϕ being the CDF of a standard normal random variable. To check the requirements of normality we use established standard tests, such as Lilliefors test ((Lilliefors, 1967), needs a large sample), Anderson-Darling test ((Anderson and Darling, 1952), for small to medium sample size, e.g. 10-200), Shapiro-Wilk test (Shapiro and Wilk, 1965), or the Jarque-Bera ((Bera and Carlos, 1980), highly attentive to outliers) test.

(2) If T_n is not a normal distribution the *Pivotal Intervals* method can be used. A $1-\alpha$ bootstrap pivotal confidence interval can be calculated by:

$$C_n = \left(2 \cdot \hat{\theta}_n - \theta^*_{((1-\alpha/2),B)}, 2 \cdot \hat{\theta}_n - \theta^*_{((\alpha/2),B)}\right)$$

where $\theta^*_{\beta,B}$ is the β sample quantile of $(\hat{\theta}^*_{n,1}, \ldots, \hat{\theta}^*_{n,B})$ and $\hat{\theta}_n = T(X_1, \ldots, X_n)$

Model Validation by Cross-Validation

Cross-validation is one of several approaches to estimating how well the model just learned from some training data is going to perform on future (yet unseen) data. It is better than the widely used *residuals approach.* The problem with residual evaluation is that it just gives a indication on how well a model fits the *given* data opposed to the a predictions for the performance on data it has not already seen. Other (more complex) method include *Akaike Information Criterion* (AIC, asymptotically equal to CV with $k = n - 1$) or *Bayesian Information Criterion* (BIC, asymptotically equal to CV with $k \approx 10$).

Cross-validation (Stone, 1974) partitions the original sample into (two or many) subsets. The analysis (e.g. model parameter estimation) is initially performed on one of these subsets (often denoted *training set*), while the other subsets (*test sets*) are used to confirm and validate the initial analysis.

A widely used method is the so called *k-fold* cross-validation. Here, the original sample is partitioned into k sub-samples. The cross-validation process is then repeated k times: in each step one of the k sub-samples (each sample is used exactly once) is used as the test set and the remaining $k-1$ sub-samples as the training set. The final results is usually computed by taking the average from the k single results.

4.3 Study Results

This section will explain the results of the algorithmic pipeline described in the previous chapter when analyzing data such as introduced in section 4.1. The outcome of the first stage of the analysis pipeline are lists of peaks that occur in a significant portion of a group at the same m/z value. These peaks are the basis for further analysis stages that yields three distinct classes of results:

Correlations: If patient meta-data are available (such as age, weight, blood parameters etc., see for example Figure 4.3.2) (cor-)relations can be sought for between peak properties (such as height) and meta-data properties (see section 4.3.1).

Fingerprints: Peaks of two groups (e.g. cancer vs. healthy) are compared to find peaks having the same m/z value in both groups but differ in properties. More formally, peaks in group A at position X that are similar

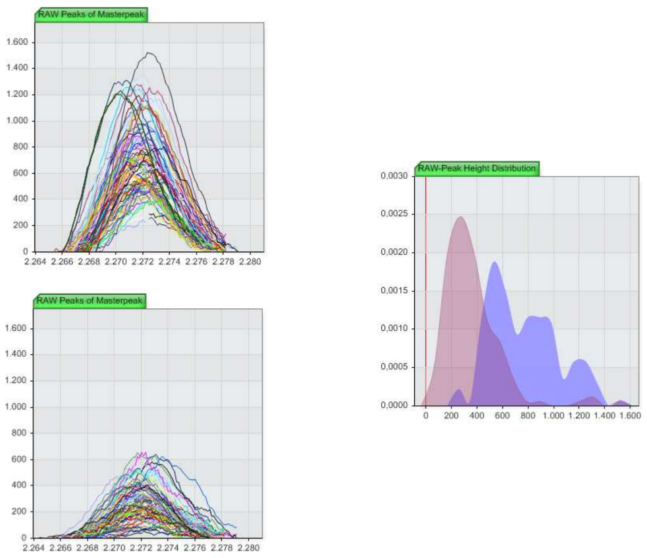


Figure 4.3.3: Left: Masterpeaks of two groups (top: men; bottom: woman) at m/z 2274da. Obviously the average height of a peak in the woman's group is lower than in the men's group. This is also clearly shown in the chart of the height distributions (right): the two distributions do overlap between 400-800, but outside this area there is a clear domination of one group.

with respect to a property P but differ to peaks in group B at position X (again with respect to property P) are sought for. An example is shown in Figure 4.3.3: two masterpeaks and their respective height distributions (of their single peaks) are shown. The best combinations of these distinguishing peaks are then used to create fingerprints. Thus, a fingerprint is basically a list of m/z values. Each list entry has attached the considered property (e.g. peak height or peak width) and the average value of this property for each group. A fingerprint can be used to classify unknown spectra (see Sec 4.3.2).

Difference Analysis: Based on the distinguishing features found during the fingerprint detection stage, further investigations can be done that analyze these differences. For example, using tandem mass spectrometry, the underlying proteins can be determined (see section 4.3.4).

Clustering: Spectra of one groups can be clustered using the fingerprint features to detect sub-groups within a group, e.g. a particular sub-group of cancer type X (see section 4.3.3).

4.3.1 Correlation of Patient Meta Data to Peaks

This analysis finds correlations of certain meta-data such as blood parameters to peak properties. These correlations are found between properties of peaks (e.g. height) from a particular group of patients (e.g. 20 year old men from a blood donator study) and available meta-data for these patients (e.g. blood parameters such as FT3).

The actual analysis computes the correlation $corX_{i,j}$ between a peak property i (e.g. peak height, peak area, peak width, peak shape) and some parameter j which can be almost anything that is related to a patient (e.g. sex, age, blood parameters, parameters from other studies concerning this patient and so forth).

We use a variety of different correlations tests such as Pearson's r (linear relation, depends on normal distributions of the variables), Spearman's R (non-linear measure, which might miss some dependencies), Kendall's τ (an improved version of Spearman's) and the Quadrant correlation, which are widely used in the applied sciences and are sensitive to different types of correlations.

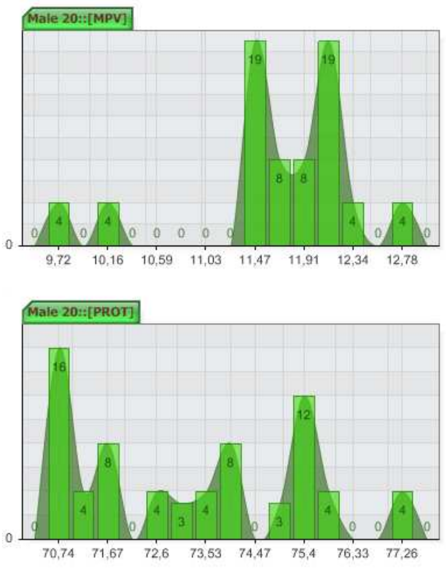


Figure 4.3.2: Distribution of two sample blood parameters (MPV, PROT) in a groups of 20 year old men.

As the next two sections will explain, results of such correlation measures have to be analyzed cautiously: for example, Pearson's test can seriously be affected by only one outlier (see e.g. (Devlin et al., 1975)). Therefore, a high correlation coefficient should be thought of as an indicator to perform further analyses, e.g. for validating the result using the cross validation schema, introduced in section 4.2.5.

Probability and Correlations

One in 20 studies using a significance level of 95% are likely to detect some statistically relevant finding by chance alone, or as (McCloskey, 1995) puts it:

> "[...] with so many people doing so many studies some of them are going to run into a one-in-20 chance of believing in a mirage."

These problems may also occur in individual studies. If a large number of statistical association between variables are analyzed (such as one masterpeak with many blood parameters), some correlations are likely to be found simply as a result of chance, even when correct statistical methods are used. To avoid this, further investigation of the relations need to be carried out, that is, it must be asked if the findings are plausible. If these questions are not asked, then there is a risk that statistics become the main quality criterion.

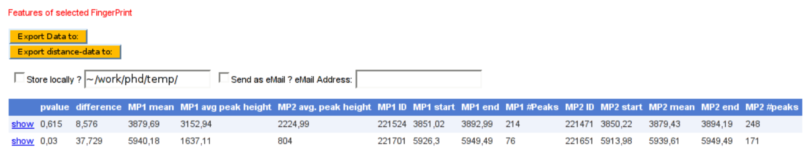
Causality

A statistical association is not the same as a causal relationship, and finding statistically significant correlations in data does not necessarily mean that causal relationships exist. As stated in the previous section, they might simply exist at random or actually based on mutual association with other (hidden) variables.

To draw conclusions about causality it is necessary to show a direct link between intervention and observed outcome. Therefore, a detected statistical association should be treated like a messenger, suggesting that further research might be valuable. Subsequently, plausibility of the correlation should be checked. Of course, plausible relations are not necessarily genuine, but it makes far more sense to use clinical or biological experience to decide what is likely to be true than to rely on statistical evidence alone.

4.3.2 Fingerprints for Classification

As mentioned in the previous sections, fingerprints are the main results of the analysis pipeline. They reflect the differences between two classes of patients



Features of selected FingerPrint

☐ Store locally ? ~/work/phd/temp/ ☐ Send as eMail ? eMail Address:

	pvalue	difference	MP1 mean	MP1 avg peak height	MP2 avg. peak height	MP1 ID	MP1 start	MP1 end	MP1 #Peaks	MP2 ID	MP2 start	MP2 mean	MP2 end	MP2 #peaks
show	0,615	8,576	3879,69	3152,94	2224,99	221524	3851,02	3892,99	214	221471	3850,22	3879,43	3894,19	248
show	0,03	37,729	5940,18	1637,11	804	221701	5926,3	5949,49	76	221651	5913,98	5939,61	5949,49	171

Figure 4.3.4: Sample fingerprint with two features (masterpeaks) at m/z positions $a = 3879.69$ and $b = 5940.18$. The table shows the details for the two masterpeaks. Not that the average peak heights differ (so the two groups can be distinguished) while the means are almost the same (the alignment did work).

(e.g. between class $c1$: "men" and class $c2$: "women") with respect to peak properties. These differences are called features (see figure 4.3.3 for an example). Figure 4.3.4 shows an example fingerprint with two features (peaks) at m/z positions $a = 3879.69$ and $b = 5940.18$ that differ in height ($c1_a = 3152$ vs. $c2_a = 2224$ for feature a and $c1_b = 1637$ vs. $c2_b = 804$ for feature b, respectively).

Recall that significance of a feature was calculated by distribution differences (see section 3.7). A second quality criterium in the fingerprint evaluation is the average effect size (section 4.2.5) for each feature, based on the mean value for each of the two distributions.

This fingerprint can now be used to classify an unknown spectrum S. A naive approach would simply find peaks in S at positions a and b, and compare the height h_a of the peak found at position a to $c1_a$ (height for the feature a in class 1) and $c2_b$ (height for feature b in class 2). The closer h_a is to one of the two values the more likely that S is of this class. Then h_b is compared to $c1_b$ and $c2_b$. If both comparisons result in the same class membership, S is assigned this class. Otherwise S is marked as *unknown*.

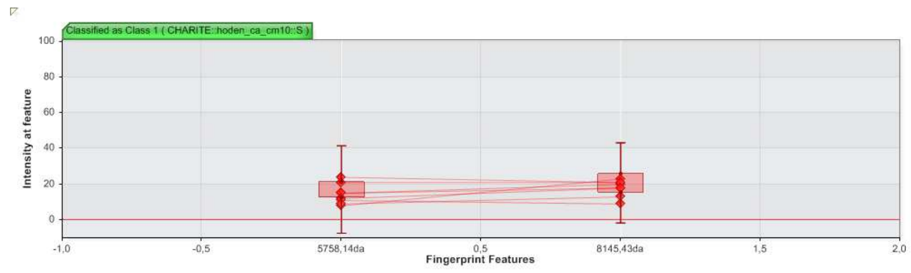
Figure 4.3.5 visualizes this procedure: The green and red vertical bars reflect the value range of each feature in the respective green or red class. The boxes within this range show the 20% range around the average value (midpoint). Diamonds reflect the actual values (here peak heights) of a feature in a particular spectrum and lines connect the feature values for the same spectrum. For visualization purposes the lines are also dyed in the color of the class represented.

The example given in Figure 4.3.5 shows the classification of a set of spectra containing two classes of patients, "class 1" and "class 2". The charts show spectra classified as "class 1" (top) and "class 2" (middle) - the bottom chart is a merge of the upper two.

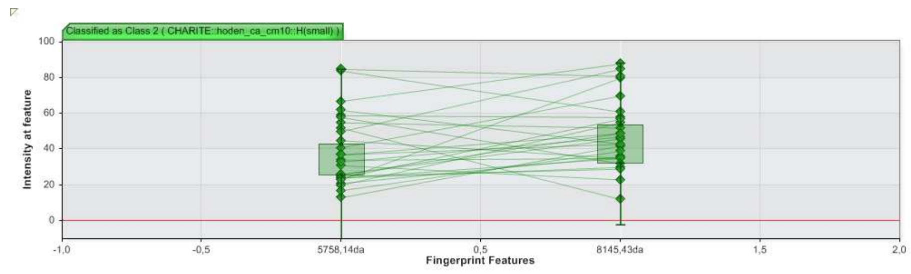
Although the naive implementation does work out, it has severe limitations especially in higher dimensions (that is many features) or if features are of different quality and should be weighted accordingly. Therefore, in the *proteomics.net* framework we have implemented the following classification algorithms using the WEKA toolkit described in (Witten and Frank, 2005).

- *SVM*: Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier. (For more details see (Platt, 1999; Keerthi et al., 1999).)

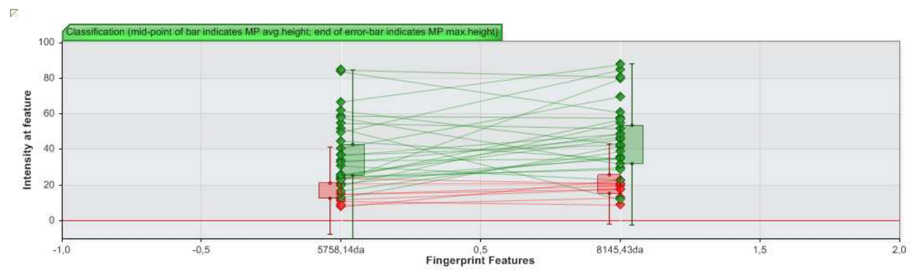
- *ANN*: Multilayer Perceptron Network (Artificial Neuronal Network) trained by the backpropagation algorithm.



(a) Classification of class 1.



(b) Classification of class 2.



(c) Classification of both classes.

Figure 4.3.5: A screenshot of a classification result. The green and red vertical bars reflect the value range of each feature in the respective green or red class. The boxes within this range show the 20% range around the average value (mid-point). Diamonds reflect the actual values (here peak heights) of a feature in a particular spectrum and lines connect the feature values for the same spectrum. For visualization purposes the lines are also dyed in the color of the class represented.

- *kNN (gen.)*: Nearest-neighbor-like algorithm using non-nested generalized exemplars. (For more details see (Martin, 1995).)
- *ADTree*: Class for generating an alternating decision tree. The basic algorithm is based on: (Freund and Mason, 1999).
- *Rand. Forest*: Class for constructing a forest of random trees. (For more details see (Breiman, 2001).)
- *Bayes Net*: Bayesian Network learning using the K2 hill climbing search algorithm.

As mentioned earlier, other algorithms (more recent and/or complicated) could have been used as well. The main reason for using more established algorithms is that they are well studied and their properties are well understood.

Each of these classification technique builds a particular model and optimizes the structure and parameters of this model using the 10-fold cross-validation approach (see section 4.2.5). If the optimal structure / parameter set is found, this classifier is evaluated on the full data-set. This evaluation results in a quality value stating how many spectra were classified correctly. To ensure a statistical relevance we also create confidence intervals as described in section 4.2.5.

4.3.3 Fingerprints for Clustering

The previous section introduced classification of unknown spectra based on fingerprint features. That is, based on a given set of peaks the classification algorithm assigns a probability value to each possible class the spectrum might stem from (e.g. "female" or "male"). The class having the highest probability values is then assigned to the spectrum.

An implicit assumption here is that all spectra having assigned the same class have very similar features. While this is generally the case, further analyses show that a class can often be divided in smaller sub-classes, based on their feature similarity. This *clustering* approach often reveals groups of patients that have other common properties not explained by the actual *class membership*, such as age class, disease *status* or other medical treatment influencing blood proteins. Thus, we want to discover something about the nature of the class from which a sample arose - to discover whether the overall class is, in fact, heterogeneous.

The next sections will first introduce some notations and presents a short overview of two widely used clustering methods, with a special emphasis on structural properties and implicit mathematical assumptions intrinsic for each of the methods. Subsequently, we show that for clustering high-dimensional spectra low-dimensional fingerprints are needed to achieve meaningful results.

Cluster Distance Functional

Let $x \in X \subset \mathbf{R}^n$ be the observed n-dimensional data. In our case x would be a spectrum's fingerprint, that is, a vector containing heights of n particular peaks. Our main goal is to partition data into $\mathbf{K}$ distinct groups (clusters) characterized by $\mathbf{K}$ distinct sets with a-priori unknown *cluster parameters*

$$\theta_1, \ldots, \theta_{\mathbf{K}} \in \Omega \subset \mathbf{R}^d, \tag{4.3.2}$$

(d is the dimension of a cluster parameter space) such that members of a particular cluster are *close* to this cluster and *far away* from other cluster. It is obvious that such methods hinge on the notation of *distance.* Let

$$d(x,\theta_i) \quad : \quad X \times \Omega \to [0,\infty)\,, \tag{4.3.3}$$

be a functional describing the *distance* from the observation x to the *cluster i.* For a given *cluster distance functional* (4.3.3), under *data clustering* we will understand the problem of a function $\Gamma(x) = (\gamma_1(x), \ldots, \gamma_{\mathbf{K}}(x))$ called the *cluster affiliation* (or the *cluster weights*) for a datum x together with cluster parameters $\Theta = (\theta_1, \ldots, \theta_{\mathbf{K}})$ which minimize the *cluster scoring functional*

$$\mathbf{L}(\Theta,\Gamma) \;=\; \sum_{t=1}^{|X|}\sum_{i=1}^{\mathbf{K}} \gamma_i(x_t)\cdot d\left(x_t,\theta_i\right) \to \min_{\Gamma(x),\Theta}, \tag{4.3.4}$$

subject to the constraints on $\Gamma(x)$:

$$\sum_{i=1}^{\mathbf{K}} \gamma_i(x) \;=\; 1, \quad \forall x \in X \tag{4.3.5}$$

$$\gamma_i(x) \;\geq\; 0, \quad \forall x \in X, \quad i = 1, \ldots, \mathbf{K}. \tag{4.3.6}$$

As we will see below choice of the cluster distance functional d (4.3.3) will bias an algorithm towards finding different types of cluster structures (or shapes) in the data. To illustrate this, we might choose d such that it will favor clusters where each member is as close to the cluster center as possible. We would expect these clusters to be compact and roughly spherical. On the other hand, we could also define our d such that each cluster member is as close to another cluster member - but not necessarily to all other members or the cluster center. Clusters discovered by this approach need not to be spherical or compact, but could have some sort of sausage shape.

A large number of different score functions can be used to measure the quality of clustering and a wide range of algorithms has been developed to search for an optimal (or at least good) partition. The exhaustive approach would be to simply search through the space of possible assignments of n points to K clusters to find that one that minimizes the score. The number of possible allocations is approximately K^n - thus, with $n = 100$ points and $K = 2$ classes we would have to evaluate $2^{100} \approx 10^{10}$ possible allocations. Since this is - of course - not feasible, the next sections exemplarily introduce concepts on how to practically optimize those score functions.

K-Means Clustering

One of the most popular clustering methods in multivariate data-analysis is the so-called *k-means algorithm* (Bezdek, 1981; Höppner et al., 1999). The affiliation to a certain cluster i is defined by the proximity of the observation $x \in X$ to the cluster center $\theta_i \in X$. In this case the *cluster distance functional* (4.3.3) takes the form of the square of the simple Euclidean distance between the points in n dimensions:

$$d(x, \theta_i) = \| x - \theta_i \|^2 . \tag{4.3.7}$$

Since we have only discrete observation x_t, $t = 1, \dots, |X|$, the functional (4.3.4) gets the form

$$\sum_{i=1}^{\mathbf{K}} \sum_{t=1}^{|X|} \gamma_i(x_t) \| x_t - \theta_i \|^2 \quad \rightarrow \quad \min_{\Gamma(X), \Theta} . \tag{4.3.8}$$

K-means algorithm iteratively minimizes the functional (4.3.8) subject to constraints (4.3.5 - 4.3.6) assigning the new cluster affiliations $\gamma^{(l)}(x)$ and updating the cluster centers $\theta_i^{(l)}$ in iteration (l) according to the following formulas

$$\gamma_i^{(l)}(x) = \begin{cases} 1 & i = \arg\min \| x - \theta_i^{(l-1)} \|^2, \\ 0 & \text{otherwise}, \end{cases} \tag{4.3.9}$$

$$\theta_i^{(l)} = \frac{\sum_{t=1}^{|X|} \gamma_i^{(l)}(x_t) \cdot x}{\sum_{t=1}^{|X|} \gamma_i^{(l)}(x_t)} . \tag{4.3.10}$$

Iterations (4.3.9-4.3.10) are repeated until the change of the *averaged clustering functional* value does not exceed a certain predefined threshold value.

The complexity of the k-means algorithm is $O(K \cdot |X| \cdot L)$, where L is the number of iterations. Note that it is possible that a good cluster solution will be missed due to the algorithm converging to a local rather than global minimum of the scoring function. One way to alleviate this problem is to carry out multiple searches from different randomly chosen starting points for the initial cluster centers. This can even be taken further to adopt a simulated annealing strategy to try to avoid getting trapped in local minima of the score function.

Fuzzy c-Means Clustering

Experiments have shown that a spectrum is very unlikely to be assigned to exactly one (sub-)cluster. In most cases a spectra reflects a transient disease status between two or more extrema (e.g. some stage between *healthy* and *fully diseased*). As it can be seen from (4.3.9), this can not be represented by the *k-means* algorithm and thus geometrically overlapping clusters can not be resolved. This issue was addressed by (Bezdek, 1981) who proposed the following modification of the *averaged clustering functional* (4.3.8):

$$\sum_{i=1}^{\mathbf{K}} \sum_{t=1}^{|X|} \gamma_i^m(x_t) \| x_t - \theta_i \|^2 \quad \rightarrow \quad \min_{\Gamma(X), \Theta} , \tag{4.3.11}$$

where $m > 1$ is a fixed parameter called the *fuzzyfier* (Bezdek, 1981; Bezdek et al., 1987). Analogously to *k-means*, the *fuzzy c-means algorithm* is an iterative procedure for the minimization of (4.3.11)

$$\gamma_i^{(l)}(x) = \begin{cases} \frac{1}{\sum_{k=1}^{\mathbf{K}} \left(\frac{\|x-\theta_i^{(l-1)}\|^2}{\|x-\theta_k^{(l-1)}\|^2}\right)^{\frac{1}{m-1}}} & \text{if } \mathbf{I}_x \text{ is empty,} \\ \sum_{r\in\mathbf{I}_x} \gamma_r^{(l)}(x) = 1 & \text{if } \mathbf{I}_x \text{ is not empty}, i \in \mathbf{I}_x, \\ 0 & \text{if } \mathbf{I}_x \text{ is not empty}, i \notin \mathbf{I}_x, \end{cases} \tag{4.3.12}$$

$$\theta_i^{(l)} = \frac{\sum_{t=1}^{|X|} \gamma_i^{(l)}(x_t) \cdot x_t}{\sum_{t=1}^{|X|} \gamma_i^{(l)}(x_t)}. \tag{4.3.13}$$

where $\mathbf{I}_x = \{p \in 1, \ldots, \mathbf{K} \mid \| x - \theta_p^{(l-1)} \|^2 = 0\}$ (Höppner et al., 1999). As it follows from (4.3.12), for any fixed *fuzzifier* m, the *cluster affiliations* $\gamma_i^{(l)}(x)$ can take values between 0 and 1, for $m \to \infty$ $\gamma_{i(x)}^{(l)} \to \frac{1}{\mathbf{K}}$. This feature allows clustering of overlapping data. However, the results are quite dependent on the choice of the *fuzzifier* m and there is no mathematically sound strategy on how to choose this parameter. Moreover, it is not clear a-priori how many clusters exist in the data and thus which value $\mathbf{K}$ should be initialized with.

Another problem not solved by this extensions is that the data is assumed being *(locally) stationary*, that is, that the *conditional expectation values* θ_i calculated for the respective clusters i are assumed to be *time independent*. This can result in misinterpretation of the clustering results, if the data has indeed a *temporal trend*, which would be the case when time series data is analyzed.

Problems Clustering High Dimensional Data

Most real-world data sets (such as a spectrum) are very high-dimensional. However, the performance of clustering algorithms tends to scale poorly as the dimension of the data grows. This is often referred to as the *curse of dimensionality* which seems to be a major obstacle in the development of data mining techniques in several ways.

In (Beyer et al., 1999; Hinneburg et al., 2000) the authors show that under certain reasonable assumptions on the data distribution in high dimensional space all pairs of points are almost equidistant from one another for a wide range of distributions and distance functions. In this paper the authors proved that the difference between the nearest and the farthest data point to a given query point (e.g. a cluster center) does not increase as fast as the distance from the query point to the nearest points when the dimensionality goes to infinity. In other words, the ratio of the distances of the nearest and farthest neighbors to a given target in high dimensional space is almost 1.

In such a case, the evaluation of the distance functional becomes ill defined, since the contrast between the distances to different data points does not exist. Thus, even the concept of proximity may not be meaningful from a qualitative perspective: a problem which is even more fundamental than the performance degradation of high dimensional algorithms.

Parameterizing Distance Metrics

As indicated in the previous section the main problem of clustering high-dimensional data is the number of dimensions used in the cluster distance functional (Equation 4.3.3). This distance metric $d(x, y)$ is a function over pairs of objects x and y from some set X. It needs to have the following properties for all $x, y, z \in X \subset R^n$:

$$d(x,x) = 0,\ d(x,y) = d(y,x),\ d(x,y) \geq 0,\ d(x,y) + d(y,z) \geq d(x,z)$$

$$d(x,y) = 0 \Rightarrow x = y$$

Commonly, d has the following form:

$$d_{A,W}(x,y) = \sqrt{(x-y)^T AWA^T (x-y)} \tag{4.3.14}$$

where $x, y \in R^n$, A can be any real matrix and W is a diagonal matrix with non-negative entries - this corresponds to a metric that “weights” the axes differently; more generally, W parameterizes a family of Mahalanobis distances over R^n. Note that AWA^T is semi-positive definite and thus $d_{A,W}(x,y)$ is a valid distance metric.

The parameterization of A and W is very flexible. For example, setting $A = I$ (the identity matrix) would result in a weighted Euclidian distance $d_{I,W}(x,y) = \sqrt{\sum_{i=1}^n W_{ii}(x_i - y_i)^2}$. Setting $A \neq I$ corresponds to applying a linear transformation to the input data $(A^T x, A^T y)$[3]:

$$d_{A,W}(x,y) = \sqrt{((x-y)^T A)W(A^T(x-y))} \tag{4.3.15}$$

Thus, we need to find a parameterization of A and W that allows rescaling of the data such that only *important* dimensions are considered in the distance calculation.

Clustering Spectra

Now we are coming back to our original problem: clustering spectra to detect biologically interesting sub-clusters e.g. within the group of spectra of a particular disease. The previous sections gave a good explanation why clustering of whole spectra with tens of thousands of dimensions would not be very reasonable. For the above reason the dimensionality of data sets is often reduced by various techniques before it is clustered.

Figure 4.3.6 shows the strategy we are using to allow clustering of very high-dimensional data. As described in sections 3.7 & 3.8.3, in this thesis we have developed a new algorithm to learn a metric that allows low-dimensional representations of high-dimensional mass spectra. This low-dimensional representation can then be used in subsequent steps such as clustering. The studies presented below (sections 4.4 & 4.5) show that our algorithms produce good results. The following steps briefly summarize our algorithm:

1. We begin with the original spectrum after **preprocessing** has been performed (see section 3.3). A raw spectrum typically has about 100.000 data points.

2. The **peak detection** step (see section 3.4) eliminates noise that “looks like” peaks and identifies biologically relevant peaks. This steps detects about 2000-4000 peaks per spectrum.

[3]Note that by introducing a non-linear basis function ϕ and using $\sqrt{(\phi(x) - \phi(y))^T W(\phi(x) - \phi(y))}$ a non-linear distance metrics could also be used.

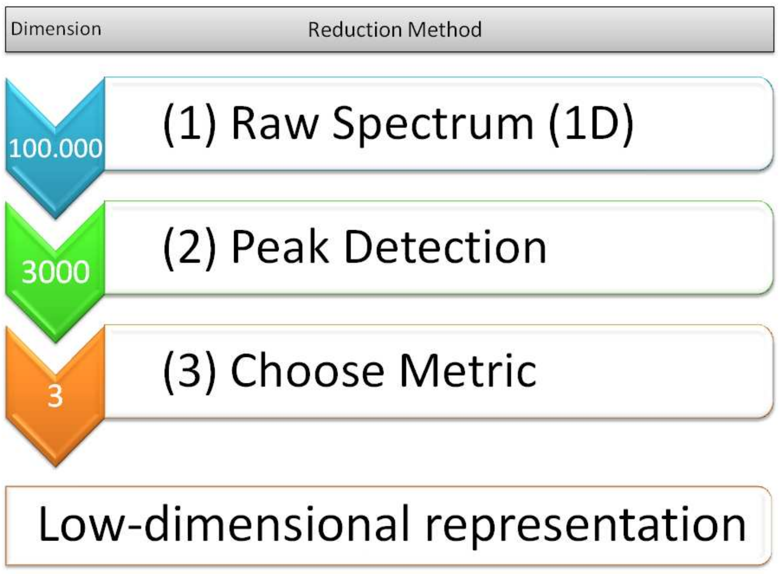


Figure 4.3.6: This figure shows the steps that are performed to achieve reduction of dimensionality starting with a spectrum of about 100.000 dimensions to a point in R^3.

3. **Choosing the metric** (see section 3.7) involves detecting significant signals and identifying a set of the most relevant peaks that can represent a spectrum. This information is then used to build the metric.

Commonly, the following two basic strategies for dimensionality reduction are used: (a) use a subset of relevant variables to construct the model (*variable selection*). That is, to find a subset of d' variables where $d' << d$. (b) transform the original d variables into a new set of d' variables, where again $d' << d$. Unfortunately, these techniques do not work well with mass spectrometry data due to two main reasons:

1. Different scales: The inherent scales of the variable values are quite different. However, common algorithms assume that the variable value reflects the importance of a variable which is not the case in out application.

2. Number of variables: Choosing the best combination of variables requires an exhaustive search over all possible feature combinations which is in general not feasible due to the high time complexity (see e.g. (Langley, 1994)). Although there exist a variety of random and heuristic approaches they only guarantee sub-optimal results.

4.3.4 Fingerprint Analysis

Features that can distinguish two groups of patients (e.g. healthy vs. diseased) are used to create fingerprints and classify unknown spectra (as described above) but can also be the basis for further investigations. The questions posed then is “What is the biological basis for this difference ?”. The basic approach to find answers to this consists of three steps:

(1) Identify the molecules that caused these peaks: There are two major methods for molecule identification:

- *"Peptide Mass Fingerprinting"* (Mann et al., 1993; James et al., 1993; Clauser et al., 1999) that uses the mass(es) of the (full or proteolytic) peptide as input to a database search of known proteins or
- Using the *Tandem MS* (MS^2, MS/MS) technique (Little et al., 1994; Mrtz et al., 1996; Wells and McLuckey, 2005; Hernandez et al., 2006) that generates collision-induced fragments of a peptide and analyzes these fragments to infer the original molecule either by searching a database for the resulting fragment pattern or by *de novo* sequencing the molecule.

There a numerous sophisticated algorithms available for peptide identification and we have integrated some of them into our framework (see section 6.2.3.

(2) Analyze biochemical pathways where this molecule plays a role: this and the next step require a sound expertise in the biological / medical area to draw meaningful conclusions. This step lays the foundation to understand the context this molecule works in. As (Villanueva, Shaffer, Philip, Chaparro, Erdjument-Bromage, Olshen, Fleisher, Lilja, Brogi, Boyd, Sanchez-Carbayo, Holland, Cordon-Cardo, Scher and Tempst, 2006) show, even small pathways such as protease activity can exhibit interesting insights.

(3) Find the reason for the different occurrence frequency: This step is certainly the most complex part of the analysis and can yield insights of varying complexity. This ranges from basic findings such as men and woman have different concentrations of sex specific hormones till very complex statements such as cancer patients show a very specific activity of some proteases that result in a quite complex peptide pattern of the fragmented pieces.

These findings should of course always be validated and confirmed by further biochemical tests and (eventually) clinical trials.

4.3.5 Medical Examples

The above paragraphs gave a very coarse grained overview of the steps necessary to analyze the features or fingerprints that can be found with our *proteomics.net* pipeline. The next two sections give an example of how this can be used in clinical environment.

4.4 Identification of Proteomic Fingerprints in Blood Serum by High-sensitive Bioinformatic Analysis of SELDI-TOF MS Data for Detection of Testicular Germ Cell Cancer

This study was performed in close collaboration with Dr. Romy Strenziok from the Charité Berlin.

Testicular germ cell cancer is the most common solid malignant tumor in young men. (Huyghe, Plante and Thonneau, 2007) showed that the incidence of testicular tumors has shown a steady increase in Europe over the past decades. Although numerous molecular parameters have been established as diagnostic and prognostic tools for a variety of tumor entities, testicular germ cell cancer and its molecular patterns are not well understood, and the literature addressing this important issue is sparse.

Routine clinical testing with serum markers is of high diagnostic value and should therefore be included in the clinical workup of testicular masses. However, these markers are only considered useful for follow-up checks and cannot adequately replace other diagnostic procedures like testicular palpation, ultrasound, and surgical exploration of suspicious testicular masses. The only available tumor marker for testicular seminoma, beta-human chorionic gonadotropin (beta-HCG), is elevated in approximately 18 % of the cases (Schmid et al., 1999). Thus there is clearly a need for new molecular markers.

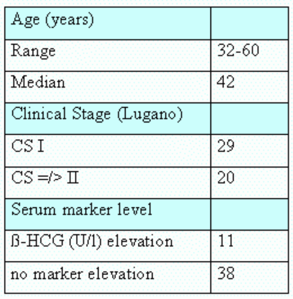
In this study we use the ProteinChip™system based on surface-enhanced laser desorption/ionization time-of-flight mass spectrometry (SELDI-TOF MS) (see section 3.2) to identify biomarkers by analyzing aberrant protein patterns in complex biological mixtures. This technology has already been beneficially applied in urological diseases such as renal cell cancer (Tolson et al., 2004) and prostate cancer (Qu et al., 2002) and for urinary protein identification in transitional cell carcinoma of the bladder (Mueller et al., 2005; Vlahou et al., 2001; Liu et al., 2005).

The aim of this study is to examine serum samples of seminoma patients by SELDI-TOF MS and assess the predictive value of this technique at primary tumor manifestation in different clinical stages.

4.4.1 Study Material

All clinical samples and data were obtained from the Charité-Universitätsmedizin Berlin, Urologische Klinik und Hochschulambulanz, Campus Benjamin Franklin, Berlin and from the Vivantes Klinikum Am Urban, Klinik für Urologie, Berlin. Serum procurement was done with the ethical approval by the Institutional Review Board.

Blood was drawn into standard serum collection tubes and allowed to clot at 4°C for 1 hour, warmed to room temperature and centrifuged for 10 minutes at 2500g. After centrifugation, all serum samples were immediately prepared and snap-frozen in liquid nitrogen before storage at -80°C. The clinically diagnosed testicular germ cell cancers were confirmed by histopathological expertising following inguinal orchiectomy. Seminoma components were immunohistologically verified by tumor-associated antigen profiling with cluster



Age (years)	
Range	32-60
Median	42
Clinical Stage (Lugano)	
CS I	29
CS =/> II	20
Serum marker level	
ß-HCG (U/l) elevation	11
no marker elevation	38

Figure 4.4.7: Total number of seminoma samples analyzed in the study (n=49).

of differentiation 30 (CD30), pancytokeratin, alpha-fetoprotein (AFP), beta-human chorionic gonadotropin (beta-HCG) and placental alkaline phosphatase (PLAP).

Healthy control samples were obtained from the Charité-Universitätsmedizin Berlin, Urologische Klinik und Hochschulambulanz, Campus Benjamin Franklin, Berlin on a voluntarily basis.

A total of 98 serum samples with 49 histologically confirmed seminoma samples and 49 age-matched controls with no history of urological disease were analyzed. The testicular germ cell cancer cohort consisted of 29 patients with localized disease (clinical stage I) and 20 patients with disseminated disease (clinical stage II) according to the Lugano classification. Marker elevation (beta-HCG) was detected in 11 seminoma patients (see Figure 4.4.7).

4.4.2 Study Description

Samples were subjected to SELDI-TOF mass spectrometric profiling using the ProteinChip System, Series 4000 Personal Edition as recommended by Ciphergen Biosystems, Inc. (Freemont, CA). Initially, various ProteinChip® chemistries were evaluated to determine which affinity chemistry provided the best spectra in terms of number and resolution of proteins. IMAC-Cu+ as well as CM10 chips gave the best results and both were used subsequently for evaluation of all serum samples.

For denaturating and fractionation of the samples we used the ProteinChip® Serum Fractionation Kit (Ciphergen Biosystems, Inc., Freemont, CA) in combination with a Biomek® 2000 Laboratory Automation Workstation (Beckman Coulter, Inc., Fullerton, CA) and followed exactly the supplied protocol. pH gradient elution (pH9/flow through, 7, 5, 4, 3, and organic solvent) resulted in 200μl fractions referred to in the following as F1 through F6. Aliquot of fractions (20μl) were bound (1:5 diluted in binding buffer) using a randomized chip/spot allocation schema to IMAC-Cu+ and CM10 (Weak Cation Exchanger) ProteinChip Arrays using a bioprocessor and the Ciphergen-recommended protocols. Finally, the energy absorbing matrix SPA (sinapnic acid, dissolevd in 250μl 1% TFA and 250μl 100% acetonitril) was manually applied as 2 x 1 μl/spot. Only fraction F4 that had been shown in preliminary experiments to give the largest number of peaks was subjected to analysis. Chips were read with the ProteinChip® Reader 4000.

Instrument settings were chosen with a laser intensity of 6000 for data shots, a mass range of 0-200000, a focus mass of 20000, a matrix attenuation of 3800, and a sampling rate of 400; 795 data shots were fired at each position. External calibration was done with a standard protein and peptide mixture.

The acquired data was then analyzed and processed by our "proteomics.net" pipeline as described in chapter 3 resulting in different sets of fingerprints for the two groups under scrutiny ("cancer" vs. "healthy").

During the analyses 138 peaks in a mass range of 3800-10000 Da where detected that could discriminate between tumor and nontumor serum samples. There was no single peak that could separate the two groups (seminoma vs. control subjects). Five peaks (6.48 kDa, 6.84 kDa, 8.14 kDa, 8.17 kDa, 8.92 kDa) were combined to construct the fingerprint to train classifiers.

4.4.3 Study Results

As the classification results show (see next section) this study clearly demonstrates that is is possible to find highly significant differences in protein profiles of cancer vs. control generated by SELDI-TOF mass spectrometry in testicular germ cell cancer patients. The potential value of the SELDI-TOF mass spectrometric pattern is particularly evident in the low molecular weight range. This feature closes the gap between SELDI-TOF mass spectrometry and valuable conventional methods like 2D gel electrophoresis that are not capable of detecting low molecular weight areas, especially in low-abundant proteins and peptides thanks to the newly developed algorithms in this thesis.

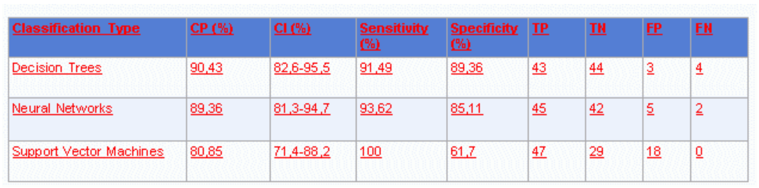
In our patient cohort, the analysis of MS-generated protein patterns not only yielded a clear assignment to either the cancer or non-cancer cohort but also enabled even beta-human chorionic gonadotropin-negative patients to be correctly distinguished from normal controls. The protein pattern on a CM10 ProteinChip® achieved 80% sensitivity and 70% specificity when classified by decision trees (95% confidence interval of 60.7%-87.1%). Four different peaks with molecular masses of 6.94 kDa, 7.76 kDa, 8.63 kDa, and 8.69 kDa were utilized to build the fingerprint and train the classifiers. In seminoma patients, beta-HCG is currently the only available tumor marker in clinical use, but according to several authors, the overall incidence of beta-HCG secretion is low and dependent on the stage of disease (Butcher et al., 1985; Huyghe, Muller, Mieusset, Bujan, Bachaud, Chevreau, Plante and Thonneau, 2007).

Patterns generated by SELDI-TOF MS would significantly facilitate the detection of marker-negative seminomas. In addition, this new proteomic approach would be most beneficial in the therapy monitoring, surveillance and follow up of these patients.

4.4.4 Classification Results

Using the fingerprint described above classifier were trained and evaluated by the 10-fold cross-validation schema. These classifier correctly predicted the spectra in 90.4% (decision tree analysis, 95% confidence interval of 82.6%-95.5%), 80.8% (support vector machines, 95% confidence interval of 71.4%-88.2%) and 89.3% of the samples (neural networks, 95% confidence interval of 81.3%-94.7%). Decision tree analysis discriminated between seminoma and healthy subjects with a sensitivity of 91.5% and a specificity of 89.4%.

Figure 4.4.8 shows the specificity and sensitivity achieved in the tested cohorts by using support vector machines and neural network analysis (CM10 ProteinChip®). Figure 4.4.9 depicts a representative spectrum of a seminoma sample in a mass range of 4000-10000 Daltons. Figure 4.4.10 shows a decision tree example generated from a CM10 ProteinChip®.



Classification Type	CP (%)	CI (%)	Sensitivity (%)	Specificity (%)	TP	TN	FP	FN
Decision Trees	90,43	82,6-95,5	91,49	89,36	43	44	3	4
Neural Networks	89,36	81,3-94,7	93,62	85,11	45	42	5	2
Support Vector Machines	80,85	71,4-88,2	100	61,7	47	29	18	0

Figure 4.4.8: Classification of the training set and test sets in 10-fold-stratified cross-validation

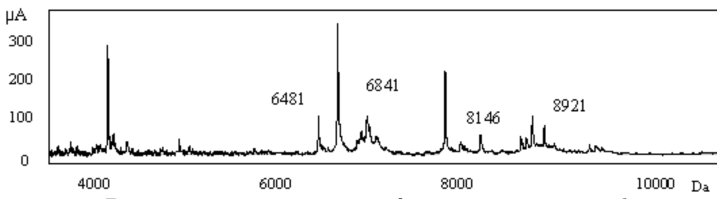


Figure 4.4.9: Representative spectra of a seminoma sample in a mass range from 4000-10000 Dalton (CM10 ProteinChip®).

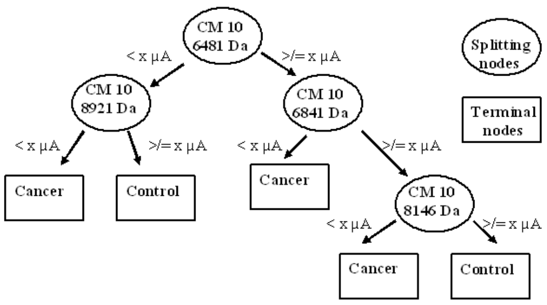
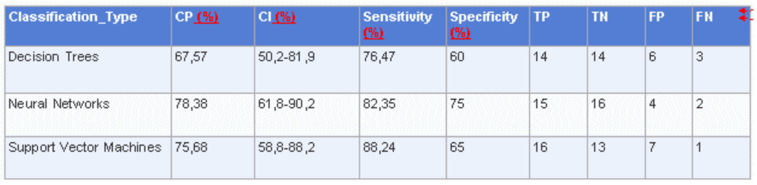
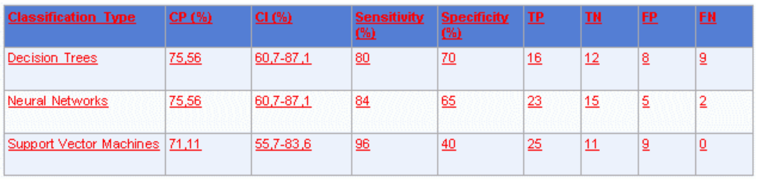


Figure 4.4.10: Decision trees generated from a CM10 ProteinChip®. Circles represent splitting nodes containing array type and moleular mass in Dalton. Peak intensity cut off is defined as < x μA (micro ampere) or >/= x μA because of different intensities (ranging between 35,1 μA-39,58 μA). Squares represent decision nodes with class assigned by algorithm (cancer and control).



Classification_Type	CP (%)	CI (%)	Sensitivity (%)	Specificity (%)	TP	TN	FP	FN
Decision Trees	67,57	50,2-81,9	76,47	60	14	14	6	3
Neural Networks	78,38	61,8-90,2	82,35	75	15	16	4	2
Support Vector Machines	75,68	58,8-88,2	88,24	65	16	13	7	1

Figure 4.4.11: Classification of the training set and test sets in 10-fold-stratified cross-validation for marker-positive seminoma patients vs. healthy controls (CM10 ProteinChip®)



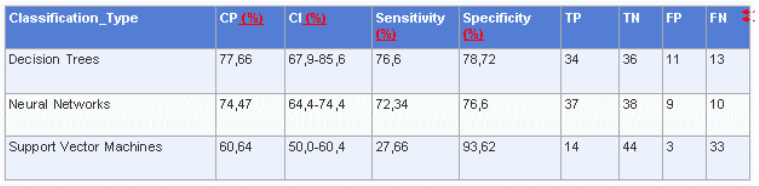
Classification_Type	CP (%)	CI (%)	Sensitivity (%)	Specificity (%)	TP	TN	FP	FN
Decision Trees	75,56	60,7-87,1	80	70	16	12	8	9
Neural Networks	75,56	60,7-87,1	84	65	23	15	5	2
Support Vector Machines	71,11	55,7-83,6	96	40	25	11	9	0

Figure 4.4.12: Classification of the training set and test sets in 10-fold-stratified cross-validation for marker-negative seminoma patients vs healthy controls, (CM10 ProteinChip®)

Assessment of beta-HCG-positive seminoma vs. control subjects yielded comparable but less significant results. Due to the lack of peaks with an adequate height difference, only three peaks were selected for analysis (CM10 ProteinChip®). The highest number of significant peaks were found in a mass range of 3800-8000 Da. Marker-positive seminomas were correctly predicted in 67.6% (decision trees, 95% confidence interval of 50.2%-81.9%), 78.4% (neural networks, 95% confidence interval of 61.8%-90.2%), and 75.8% of the cases (support vector machines, 95% confidence interval of 58.8%-88.2%). The three analytical methods had adequate sensitivities ranging between 76.5% and 88.2%. Their specificities were likewise adequate at 60% (decision trees), 75% (neural networks), and 65% (support vector machines), respectively. Three different peaks (3.80 kDa; 4.98 kDa; 7.97 kDa) were chosen to create a fingerprint.

Figure 4.4.11 displays the corresponding sensitivities and specificities. Decision trees achieved 80% sensitivity and 70% specificity in discriminating between beta-HCG-negative seminoma and control subjects.

Figure 4.4.12 shows decision trees, neural networks, and support vector machine analyses. The rates of correctly classified subjects in all groups are 71.1%-75.7% (CM10 ProteinChip ®, 95% confidence interval of 60.7%-87.1% for decision trees; 95% confidence interval of 60.7%-87.1% for neural networks; 95% confidence interval of 55.7%-83.6% for support vector machines; molecular masses: 6.94 kDa; 7.76 kDa; 8.63 kDa; 8.69 kDa). The CM10 ProteinChip ® achieved excellent differentiation between seminoma and control subjects, while the IMAC-Cu+ chip yielded significantly less precise results. The combination of 6 peaks (4.19 kDa, 6.37 kDa, 7.76 kDa, 7.93 kDa, 7.97 kDa and 9.29 kDa) predicted cancer patients in 60.6%-76.6% (decision trees, neural networks and support vector machines) with a sensitivity of 27.6% to 76.6% and a specificity of 76.6% to 93.6% (Figure 4.4.13) in the 10-fold stratified cross-validation (95% confidence interval of 67.9%-85.6%, decision trees; 95% confidence interval of 64.4% -74.4%, neural networks; 95% confidence interval of 50.0% - 60.4%, support vector machines).



Classification_Type	CP (%)	CI (%)	Sensitivity (%)	Specificity (%)	TP	TN	FP	FN
Decision Trees	77,66	67,9-85,6	76,6	78,72	34	36	11	13
Neural Networks	74,47	64,4-74,4	72,34	76,6	37	38	9	10
Support Vector Machines	60,64	50,0-60,4	27,66	93,62	14	44	3	33

Figure 4.4.13: Classification of the training set and test sets in 10-fold-stratified cross-validation for seminoma patients vs healthy controls (ProteinChip® IMAC-Cu+).

Protein profiles generated by the IMAC-Cu+ array did not differ significantly between beta-HCG-positive and beta-HCG-negative seminoma patients and healthy controls.

4.5 Identification of Proteomic Fingerprints in Blood Serum by High-sensitive Bioinformatic Analysis of MALDI-TOF MS Data for Detection of Thyroid Diseases

This study was performed in close collaboration with Dr. Alexander Leichtle from the Institute of Laboratory Medicine, Clinical Chemistry and Molecular Diagnostics at University Leipzig.

Today, thyroid diseases are very common in the general population. For example, up to one third of the adult German population suffers from nodular thyroid disease (Hampel et al., 1995; Kratzsch et al., 2005). Studies have shown that even failure of thyroid function shows a prevalence of up to 10%, whereas the presence of positive anti-TPO antibodies (TPOAb) and of positive anti-thyroglobulin antibodies (TgAb) is slightly higher in a US population (13% and 11.5%) and more pronounced in white population, females and the elderly ((Hollowell et al., 2002)). (Zphel et al., 2003) showed that TPOAb are detectable in nearly all euthyroid individuals and TPOAb values in the low measurable range are normally distributed.

Applying the *National Academy of Clinical Biochemistry* (NACB) decision limits applied to older men or women, there is a markedly increased number with "elevated" autoantibody levels compared to sex- and age-specific reference intervals (O'Leary et al., 2006). TgAb and TPOAb are of immunoglobulin G (IgG) class and have high affinities for their respective autoantigens. (McLachlan and Rapoport, 2004) have shown that both autoantibodies are markers of thyroid autoimmunity. While TgAb alone in the absence of TPOAb is not significantly associated with thyroid disease, TPOAb and the combination of both, TgAb and TPOAb, however, show strong association with clinical hypo- and hyperthyroidism (Hollowell et al., 2002).

The familial aggregation of thyroid autoantibodies seems to be mainly genetically determined (Brix et al., 2004): An exon 1 CTLA-4 gene polymorphism G allele influences higher TPOAb and TgAb production, whereas the C allele affects specifically TPOAb production in patients with Hashimoto's thyroiditis (Zaletel et al., 2006) suggesting CTLA-4 as a major thyroid autoantibody susceptibility gene.

High TPOAb titre correlates with increased frequencies of T cells producing Th/Tc1 cytokines, probably responsible for thyroid cell damage and/or death in Hashimoto's thyroiditis (Karanikas et al., 2005). TPOAb and TgAb are both correlated with thyroid enlargement (Carl et al., 2006). The histological diagnosis of Hashimoto's thyroiditis can most precisely be predicted by TgAb measurement (Kasagi et al., 1996).

(Engum et al., 2005) have found no associations between antithyroid antibodies and depression or anxiety in a population-based study. However, fibromyalgia patients had thyroid autoimmunity higher than control subjects (Pamuk and Cakir, 2007). There is also a strong relation between thyroid autoimmunity and breast cancer (Giustarini et al., 2006), but a direct relationship between thyroid autoimmunity and breast cancer seems to be unlikely (Kuijpens et al., 2005).

Pregnant women seem to have a lower positive rate of TPOAb than non-pregnant women (3.3% vs 9.4%, p ¡ 0.01) (xia Guan et al., 2006) and the prevalence of both, TPO and Tg antibodies shows a progressive decline throughout gestation becoming undetectable in the third trimester (Smyth et al., 2005).

(Radetti et al., 2006) showed that in children, the presence of goiter and elevated TgAb at presentation, together with progressive increase in both TPOab and TSH, may be predictive factors for the future development of hypothyroidism.

So far several studies using proteomic techniques have been performed in the area of thyroid diseases, mainly focusing on thyroid malignancies (Villanueva, Martorella, Lawlor, Philip, Fleisher, Robbins and Tempst, 2006; Brown et al., 2006; Suriano et al., 2006) and thyroid tissue pathology (Krause et al., 2006; Torres-Cabala et al., 2006). However, the proteomic implications of thyroid autoantibodies in the general population still remain unclear.

Due to the high clinical impact of thyroid autoantibodies even in a non-symptomatic population, this study investigates serum proteome profiles in an apparently healthy population of blood donors to detect specific discriminating proteome patterns which could very early point to the underlying autoimmune processes in subjects with high thyroid autoantibody titers.

Study Participants

We included a sub-group of 870 healthy blood donors from one study center (experienced and first time donors, which met the general principles of donor selection and guidelines for deferral according to the German Guidelines (Bundesgesundheitsblatt-Gesundheitsforschung-Gesundheitsschutz, 2000)) without a personal history of a thyroid disease.

All donors filled out a questionnaire requesting information concerning a family history of thyroid diseases, anthropometric data, smoking habits, and medications. A total of 445 male and 425 female donors (age range, 18-68 years) were randomly included.

Thyroid ultrasonography was performed with an EUB-405 (Hitachi) with a 7.5 MHz transducer. The thyroid volume was calculated as length · width · depth · 0.479 for each lobe (J et al., 1981). Goiter was defined as a thyroid volume exceeding 18 mL in women and 25 mL in men (Gutekunst et al., 1988). Solid nodules were identified as differing from the healthy thyroid tissue in pattern and ultrasonic echo intensity. The nodules were classified as isoechoic if their texture closely resembled that of healthy thyroid tissue, hyperechoic if more echogenic, and hypoechoic if less echogenic.

Venous blood was collected before the blood donation procedure started using Sarstedt blood collection systems (Sarstedt, Nümbrecht, Germany). After blood was centrifuged at 3000g for 15 min, the serum was aliquoted and stored frozen at -80°C until analysis.

All blood donors gave written consent for the participation in this study and the study was also approved by the local ethics committee.

Study Description

Serum concentrations of TSH, FT4, FT3, T4, and T3 were measured by assays on the ELECSYS system (Roche Diagnostics). The immunoreactivity of serum TPO antibodies and Tg antibodies was measured on the same platform.

After thawing on ice peptide and protein purification and fractionation was performed using a magnetic bead-based separation technique (ClinProtTM, Bruker Daltonics, Leipzig, Germany) with specific surface functionalities (MB-IMAC Cu, MB-WCX, MB-HIC C8). The procedure was performed exactly according to the recently published protocol (Baumann et al., 2005).

Gradient grade acetonitrile, ethanol, acetone were obtained from J.T. Baker (Phillipsburg, USA); trifluoroacetic acid was purchased from Sigma-Aldrich (Steinheim, Germany). Peptide and protein calibration standards, a-cyano-4-hydroxycinnamic acid were purchased from Bruker Daltonics (Leipzig, Germany). Peptide preparations were done in 0.2ml polypropylene tubes (8-tube strips) from Biozym (Hess. Oldendorf, Germany). The MALDI-TOF AnchorChip™ target (four spots per sample) was purchased from Bruker Daltonics (Leipzig, Germany).

The proteome profiling was performed using a ClinProt™ Robot and an Autoflex Linear MALDI-TOF Mass Spectrometer (Bruker Daltonics, Germany). Mass spectra were recorded by the flexControl™ 2.0 Software (Bruker Daltonics, Germany). The settings were applied as follows: Ion source 1: 20 kV; ion source 2, 18.50 kV; lens, 9.00 kV; pulsed ion extraction, 120 ns; nitrogen-pressure, 2500 mbar. Ionization was achieved by a nitrogen laser (λ=337 nm) operating at 50 Hz. For matrix suppression a high gating factor with signal suppression up to 500 Da was used. Mass spectra were detected in linear positive mode. Spectral data were combined with beforehand surveyed epidemiological and clinical meta-data to provide highly differentiated classification criteria. Overall, about 8500 spectra were acquired during data collection. The following data processing was performed according to our standardized workflow as described in chapter 3

Study Results

In our study group, gender was comparably distributed within all age groups (data not shown). Thyroid volume was significantly higher in males than in females (19.9 vs. 14.7, $p < 0.001$). Other physiologic variables such as blood pressure and heart frequency showed values and gender-dependent differences within the expected intervals for healthy persons. Approximately one-fifth of the donors (16.2% males and 26.1% females) reported a family history of thyroid diseases. Goiter and/or hyperechoic or hypoechoic areas were newly detected in 26.8% of the investigated females and 23.8% of the investigated males. Of the 870 donors recruited, 220 showed irregularities of the thyroid or had a positive family history for thyroid diseases. Of all persons examined, 9.6% had increased TPO antibody immunoreactivity and 10.3% had Tg antibody immunoreactivity; 16.0% of these showed immunoreactivity for both antibodies.

Serum samples were analyzed using magnetic-bead based MALDTI-TOF MS. After preprocessing diverging peaks for normal/high TPO and Tg antibodies were selected manually with respect to peak height and distance measure.

For TPO antibody classes (above and below 37.1 IU/mL) we build a fingerprint (FP1) with discriminating signals at 1836.0 Da (Jensen-Shannon measure of divergence [JS] 41.7), 1884.5 Da (JS 53.3) and 1732.8 Da (JS 39.6) (see Figure 4.5.14).

For Tg antibody classes (above and below 98.1 IU/mL) we built a fingerprint (FP2) with discriminating signals at 1084.5 Da (JS 60.0), 2755.4 Da (JS 51.3) and 4064.2 Da (JS 135.5) (see Figure 4.5.15).

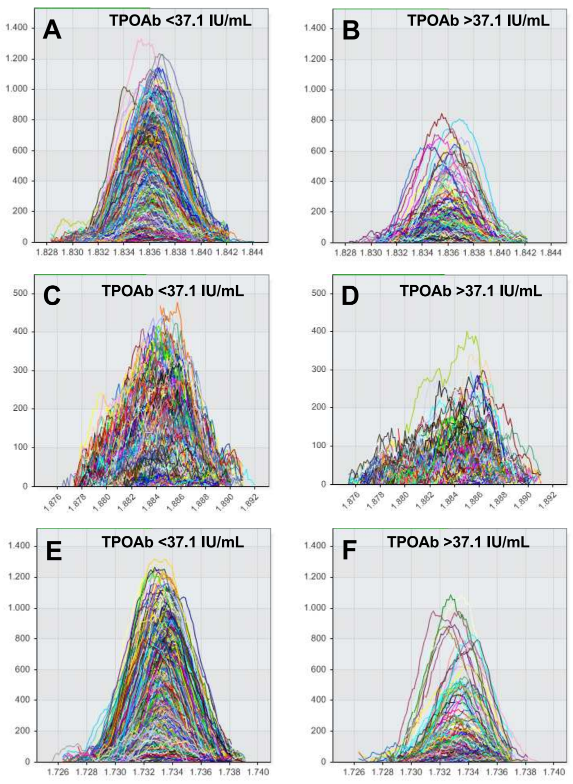


Figure 4.5.14: Fingerprint FP1 (selected signals) for TPO antibody classes.

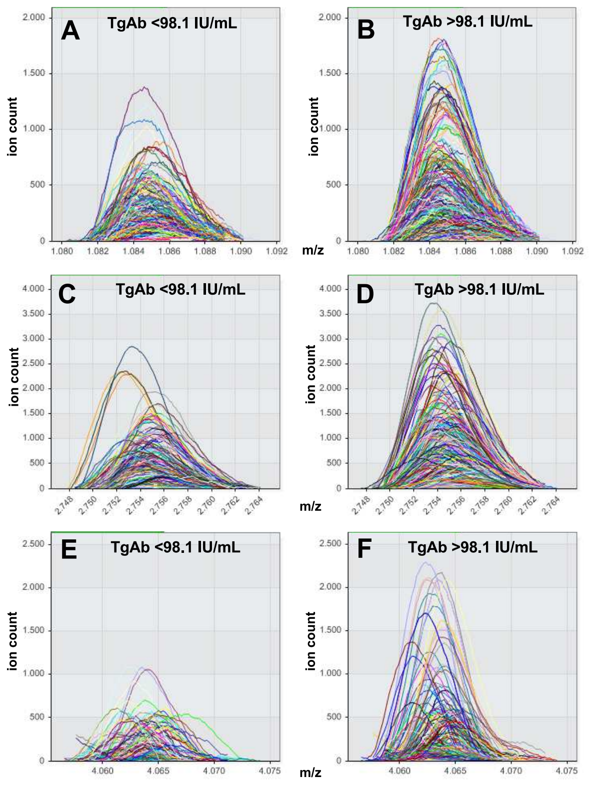
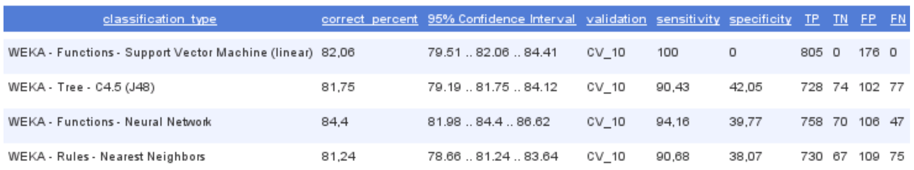


Figure 4.5.15: Fingerprint FP2 (selected signals) for the Tg antibody classes.

Training classifiers with fingerprint F1 we used 10-fold cross-validation to measure the performance of these classifiers. The results are shown in Figure 4.5.16.

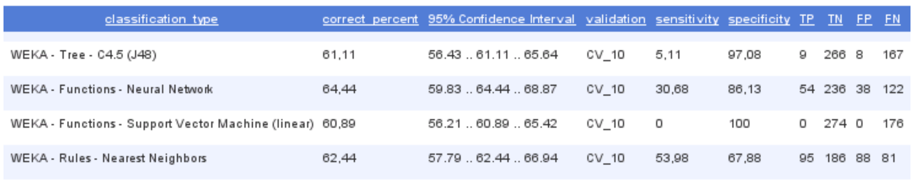
Training classifiers with fingerprint F2 we used 10-fold cross-validation to measure the performance of these classifiers. The results are shown in Figure 4.5.17.

To show that the fingerprints are generalizable we then used the fingerprints F1 and F2 to build classifiers that can distinguish between the TPO and Tg antibody classes. Performance was again measured by the 10-fold cross-validation schema. These results are shown in Figure 4.5.18.



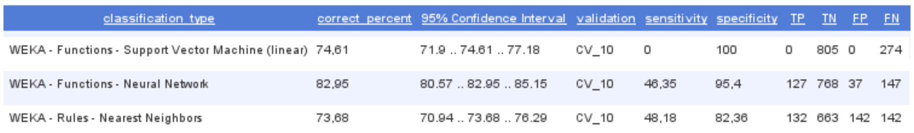
classification type	correct percent	95% Confidence Interval	validation	sensitivity	specificity	TP	TN	FP	FN
WEKA - Functions - Support Vector Machine (linear)	82,06	79.51 .. 82.06 .. 84.41	CV_10	100	0	805	0	176	0
WEKA - Tree - C4.5 (J48)	81,75	79.19 .. 81.75 .. 84.12	CV_10	90,43	42,05	728	74	102	77
WEKA - Functions - Neural Network	84,4	81.98 .. 84.4 .. 86.62	CV_10	94,16	39,77	758	70	106	47
WEKA - Rules - Nearest Neighbors	81,24	78.66 .. 81.24 .. 83.64	CV_10	90,68	38,07	730	67	109	75

Figure 4.5.16: Classification of the training set and test sets in 10-fold-stratified cross-validation. Data is shown for four different classifiers using the fingerprint F1 after classification of $n = 981$ spectra "above 37.1 IU/mL TPO antibody" vs. "Control".



classification type	correct percent	95% Confidence Interval	validation	sensitivity	specificity	TP	TN	FP	FN
WEKA - Tree - C4.5 (J48)	61,11	56.43 .. 61.11 .. 65.64	CV_10	5,11	97,08	9	266	8	167
WEKA - Functions - Neural Network	64,44	59.83 .. 64.44 .. 68.87	CV_10	30,68	86,13	54	236	38	122
WEKA - Functions - Support Vector Machine (linear)	60,89	56.21 .. 60.89 .. 65.42	CV_10	0	100	0	274	0	176
WEKA - Rules - Nearest Neighbors	62,44	57.79 .. 62.44 .. 66.94	CV_10	53,98	67,88	95	186	88	81

Figure 4.5.17: Classification of the training set and test sets in 10-fold-stratified cross-validation. Data is shown for four different classifiers using the fingerprint F2 after classification of $n = 450$ spectra "above 98.1 IU/mL Tg antibody" vs. "Control".



classification type	correct percent	95% Confidence Interval	validation	sensitivity	specificity	TP	TN	FP	FN
WEKA - Functions - Support Vector Machine (linear)	74,61	71.9 .. 74.61 .. 77.18	CV_10	0	100	0	805	0	274
WEKA - Functions - Neural Network	82,95	80.57 .. 82.95 .. 85.15	CV_10	46,35	95,4	127	768	37	147
WEKA - Rules - Nearest Neighbors	73,68	70.94 .. 73.68 .. 76.29	CV_10	48,18	82,36	132	663	142	142

Figure 4.5.18: Classification of the training set and test sets in 10-fold-stratified cross-validation. Data is shown for four different classifiers using the fingerprints F1 & F2 after classification of $n = 1097$ spectra "above 37.1 IU/mL TPO antibody" vs. "above 98.1 IU/mL Tg antibody".

4.6 Biological Applications

In addition to the clinical application described in the previous sections mass spectrometry has also emerged as the preferred method for in-depth characterization of protein components of biological systems. Many studies have shown that is is possible to gain key insights into the composition, regulation and function of molecular complexes and pathways. Thus, mass spectrometry-based proteomics has evolved to a quite powerful *hypothesis-generation* platform. This tool in combination with complementary techniques from areas such as molecular biology or pharmacology provides a framework to gather more understanding of complex biological processes. This section will briefly review some recent studies and show the large range of applications.

4.6.1 Characterization of Protein Complexes

Many proteins found in cells or tissue function as components of larger complexes. These complexes can vary in size dramatically, from just a few small proteins clustering together to very large assemblies of dozens of proteins, such as the ribosome. Since the composition of these complexes is highly regulated in cells (context and time dependent) unraveling the structure becomes a real challenge. Mass spectrometry-based proteomics can aid in this issue as described in the examples below.

Discovering a Mitochondrial Protein Complex Structure

By mass spectrometry-based proteomics analysis of mitochondrial BAD-containing complexes (Danial et al., 2003) discovered an unexpected physical association between a key apoptotic protein (BAD) and a key glycolytic protein (glucokinase). This led to new models to explain how cells coordinate metabolic signals and survival signals.

Discovering a Transcription-Factor Complex

(Ranish et al., 2004) used mass spectrometry-based proteomics to characterize a previously unknown component of TFIIH (one of the transcription factors that make up the RNA polymerase II preinitiation complex) that eventually succeeded in explaining the molecular basis for a human photosensitivity syndrome.

Discovering a Chaperone Complex

In an early study (Washburn et al., 2001) analyzed the CFTR (cystic fibrosis transmembrane conductance regulator, an ABC transporter-class protein and chloride ion channel) interactome using mass spectrometry-based proteomics and identified specific co-chaperones and chaperone folding pathways that seem to control mutant channel stability, cell-surface expression and function. Mutations of CFTR leads to cystic fibrosis and congenital absence of the vas deferens (part of the male anatomy).

4.6.2 Characterization of Protein Pathways

Another very exciting application of mass spectrometry-based proteomics is the comparative analysis of biological samples of different conditions, such as in human cancer research (see above) but also in understanding a biological system. These studies usually aim at the identification of protein fingerprints that characterize the difference between a system at the different stages. Below we give some success stories in this area of mass spectrometry-based research.

Identification of Proteins in Kinase Pathways

A compelling example of the value of comparative proteomics for assigning unique cellular functions to uncharacterized members of protein classes was done by (Khan et al., 2005). They developed an innovative cell-biological strategy based on mass spectrometry proteomics to enrich distinct sexual stages of P.berghei life cycle that allowed the generation of high-quality cellular models for in-depth analysis. The result of this study was the most comprehensive inventory of sex-specific parasite proteins generated so far, including the discovery of novel protein kinasis regulating sex-specific signalling pathways.

Identification of ATM/ATR candidates in DNA-damage Response Pathways

An excellent example how quantative mass spectrometry-based methods can used for mapping protein phosphorylation sites in proteomes is given by (Matsuoka et al., 2007).

Identification of Proteins in Insulin Pathways

(Dong et al., 2007) show that quantitative mass-spectrometry-based proteomics using stable isotope labelling can be applied to intact organisms, as well as cell-culture preparations, thus greatly expanding the potential applications of this method.

4.6.3 Fingerprinting the Influenza Virus

Influenza is still a deadly virus that continues to cause illness all over the world. The Influenza type A virus (most virulent in humans) genome is contained on eight single (non-paired) RNA strands that code for eleven proteins. One of these proteins, HA, encodes hemaglutinin which determines the extent of an infection into a host organism. Screening of the virus is necessary for the development of effective vaccines, since even a point mutation in the HA gene sequence can render a vaccine ineffective. Usual screening techniques such as enzyme immunoassays, complement fixation and heagglutination inhibition (HI) assays are quite time consuming and do not provide molecular details.

A recent paper by (Downard and Morrissey, 2007) introduces a new surveillance approach for screening the structure and antigenicity of the virus. The method is based on creation of MS and MS/MS spectra from the tryptic digest of hemagglutin. The resulting MS fingerprints can be compared to reference fingerprints and by analyzing the MS/MS spectra even the molecular structure and sequence can be determined in a single step.

Chapter 5

Computer Science Grid Strategies

Contents

The previous chapters have introduced new methods for the analysis of mass spectrometry data. The main advantage of our algorithms was increased sensitivity that - unfortunately - introduced more complex computations and increased amount of data. To speed up these calculations we developed a new framework that can split up the analyses tasks and distribute these sub-tasks to compute machines organized in an environment we call the *quasi ad-hoc Grid.* This framework and its advantages (e.g. over commonly used compute clusters) are described in the following sections.

5.1 Introduction

Driven by increasingly complex problems, new technologies and machines producing gigabytes of data each day, today's science is based on computation power and data analysis as never before. But even as development of computer power and data storage continue to improve exponentially[1], these resources are failing to keep up with what scientists demand of them. As an example, scientists back in 1990 were happy assembling small parts of DNA sequence information of a chromosome. These days they want to assemble the complete human genome and several physics projects such as CERN's Large Hadron Collider, produce multiple petabytes of data per year. Of course, current desktop computers are much more powerful than supercomputers in the early 1990s and the storage a PC ships with is comparable with an entire 1990

[1]According to *Moore's Law* stating that semiconductor power doubles roughly every 18 months.

supercomputer center, but from these two examples it gets obvious that the demand will always be greater than what current technologies can deliver.

The typical approach to tackle this issue is what computer scientists call *divide and conquer*: break down the original problem into smaller sub-problems and let them be solved by many processors in parallel in (much) shorter time. Then the solutions to the sub-problems are combined to give a solution to the original problem. The actual computation of these sub-problems is usually done by using one of the following concepts:

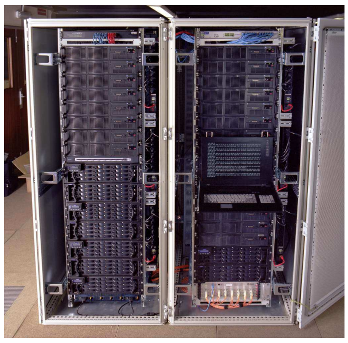


Figure 5.1.1: A typical cluster as it is used in many computing centers. The picture shows a server rack with computing, storage and control units which are connected through a fast backbone network.

- Using a *Supercomputer* consisting of hundreds or thousands of processors connected by an ultra-fast network and often accessing shared memory. Therefore, data exchange within this machine is extremely fast. A supercomputer costs from hundreds of thousands to tens of millions of Euros. Examples are NEC's "Earth Simulator" or IBM's Blue Gene/L with over 131.000 processors.

- Using a *computer cluster*. A cluster is usually a (fixed-sized) group of identical[2] computers connected by a fast network (see Figure 5.1.1) and controlled by a central host (master / slave approach). The supposedly first cluster was built at a NASA research center in 1994 and consisted of 16 Intel 486 PCs. The big advantage of a cluster system over a classical supercomputer is the price tag. While performing equally well with respect to speed it is usually 50 - 90% cheaper to build a cluster than buying a supercomputer, because a cluster can be built from cheap commodity hardware. The biggest downside is that communication between two nodes takes some hundred microseconds opposed to just a few microseconds within a supercomputer.

- Using a *computing Grid*. The Grid is a type of Infrastructure similar to a cluster. The main difference is that a Grid has no fixed size - its nodes are loosely connected together dynamically on demand. The probably most interesting advantage of a Grid system over the supercomputer and the cluster is that every computer can be part of a Grid no matter where it is located or if other users are working on this machine. That is, a Grid can be build with any existing hardware so basically no money needs to be spent!

Since the cluster and the Grid approach are much cheaper while delivering the same computational power we will abandon the supercomputer idea. It should be noted however, that if sub-problems need to communicate with each other frequently (e.g. because sub-problems are very quickly computed) a supercomputer is the first choice. This is because the other concepts need (comparatively) quite a long time for communication between nodes.

Comparing a cluster system and the Grid approach, the key differences and main advantages of Grids are:

- Each node can be any piece of computer hardware opposed to a cluster where (usually) they need to be of the same type of hardware and run the same operating system.

[2]Identical here means that the hardware, operating system and installed software is the same.

- A node is not exclusively used for the Grid whereas in a cluster it is completely used by the cluster Application - it follows that even office computers or laptops can be integrated into a Grid.
- A node can be at any physical location in the whole world as long as it can connect to the Grid (e.g. by the Internet) whereas they need to be in the same local network in the cluster case and use the same access policies (for control reasons).
- Nodes can be added dynamically on demand opposed to a cluster that usually has a fixed size (until new nodes are bought and manually integrated).

In this project we have focused on the Grid approach. In a university setting this follows almost automatically from the fact that there is usually a large number of (heterogeneous) computing resources available within the university or other partner institutes. The resources are idle most of the time. With the Grid approach resources can be added (when they are unused) or excluded (when a user starts using them) from the Grid dynamically and even resources from an existing cluster can be integrated.

However, as pointed out in section 5.1.3 using the power of Grid computing these days is still very complicated. This is due to quite complex Grid frameworks needed to be set up and used for the computations. There is still no easy and quick way to set up such a system crossing organizational borders to work mutually on a problem. Projects such as SETI@HOME (see section 5.7) have made a remarkable step towards this direction but they are highly specialized on one particular problem domain. To the best of our knowledge there is still nothing that allows users to set up a Grid spontaneously and compute any (suited) large problem on many different resources in an easy way.

In this thesis we have developed an approach we call the *Quasi ad-hoc Grid* fulfilling these requirements. Additionally to the commonly available functions of a Grid platform, namely

- easily set up a Grid Infrastructure with distributed heterogeneous resources
- integrate thousands of working nodes working mutually together on (suited) large scale problems (opposed to systems like the SETI@HOME project where only one particular problem can be analyzed)

it allows users to

- do this almost instantaneously (ad hoc) without the need of installing client software prior to computation (see section 5.3.3 on page 126)
- use almost any programming language for the computations - even Matlab™(see section 6.2.5 on page 159)
- use almost any hardware platform - even Sony's Playstation 3™(see section 6.2.4 on page 158)
- hot deploy new services into an existing structure (see section 5.5.4 on page 140)

- create complex workflows to be executed on the Grid (see section 5.1.2 on page 109)

It is a *quasi* ad hoc approach since we need a central management entity usually not necessary in ad hoc networks.

5.1.1 The History of Grid Computing

The term Grid computing was coined by Ian Foster and Carl Kesselmans (Kesselman and Foster, 1998) in the early 1990s as a metaphor for making computer power as easy to access as an electric power Grid. It is often ambiguously used with "distributed computing", "High Performance Computing (HPC)", or "Virtual Supercomputing".

The concept of sharing distributed resources is actually quite old. In 1965, Corbat'o and his team envisioned a computer facility operating "like a power company or water company" (Vyssotsky et al., 1965), and in (Licklider and Taylor, 1968) the authors anticipated Grid-like scenarios. Since the late 1960s, much work has been devoted to developing distributed systems, but with mixed success.

Taking off in the late 1970s (virtual) supercomputers were initially limited to Applications in defense industry and a small number of high technology industries. Since the early 1990s, however, Grid computing entered other sectors of business, industry and science. Computer manufactures have understood the demand of the market and responded by developing many different computer architectures ranging from traditional vector processors to assemblies of RISC processors [3] . Nowadays, Grid computing has been accepted by the global computing community and is used successfully for solving large scale problems in industry, business and science.

Important Grid projects at the moment are Folding@Home (Stanford University; the goal is to understand why proteins misfold), Seti@Home (UC Berkeley; they search for signs of extra-terrestrial intelligence), or Evolution@Home (addressing fundamental questions about evolution and population genetics). Each project is usually running their own Grid platform such as BOINC (Anderson, 2004), Globus (Foster and Kesselman, 1997), GridBus, Condor (M. J. Litzkow and Mutka, 1988) (which is actually a scheduler), Beowulf (Sterling et al., 1995) (which is actually a cluster concept) or the SUN Grid Engine (Gentzsch, 2001) (which is actually a scheduler). The Globus toolkit has evolved into something the New York Times calls the *de facto standard* for Grid computing.

5.1.2 Grids at a Glance

Traditionally, a Grid consists of three main components: (a) the Compute Grid where the actual problem is computed, (b) the Data Grid providing the data and (c) the (optional) Grid Portal that allows easy submission to, control

[3]The reduced instruction set computer, or RISC, is a CPU design philosophy that favors an instruction set reduced both in size and complexity of addressing modes, in order to enable easier implementation, greater instruction level parallelism, and more efficient compilers. As of 2007, common RISC microprocessors families include the DEC Alpha, ARM, Power Architecture, PowerPC and SPARC. The idea was originally inspired by the discovery that many of the features that were included in traditional CPU designs to facilitate coding were being ignored by the programs that were running on them. Also these more complex features took several processor cycles to be performed.

of and configuration of the Grid. However, when using a Grid some concepts known from traditional single-machine oriented problem solving cannot be applied anymore. Some key points to be reconsidered are listed below and will be discussed in the remaining of this chapter:

- Problem splitting and result merging: Obviously the original problems needs to be decomposed into hundreds, thousands or even millions of tasks, each of which is executed (nearly) independently. The most common decomposition approach exploits a problem's inherent data parallelism - breaking the problem into pieces by identifying the data subsets, or partitions, that comprise the individual tasks. These and the corresponding data partitions are then distributed to the compute nodes for processing. It is obvious that for each single problem the right strategy needs to be found. This feature is commonly not directly implemented in the Grid platform but in a so-called *scheduler*. After the computations are finished they need to be merged together to form the final result.
- Data transportation: Sub-problems and their associated (often very large) data, results and status messages must be communicated through the Grid efficiently and reliably.
- Inhomogeneous nodes: A computer program is no longer written for a particular system; instead, a Grid node can be almost any combination of hardware, operating system and programming languages (or their respective interpreters). To allow integration of special hardware (like Sony's Playstation) and software (such as Matlab) special client software adapted to the local environment is needed.
- Changing conditions: Since (mostly) nodes are not exclusively used for the Grid job executing control must be performed and events such as loss of nodes must be handled.
- Restricted access: Usually a user of a Grid has no administrative access to all machines. This affects the process of distributing and running the client software on the nodes.
- Reliability of Results: The results should be double-checked since computers which are performing the calculations might not be entirely trustworthy.
- Security: Neither the machines itself nor the communication within the Grid is fully trustworthy.

Different Types of Grids

Since there are different groups of users having different needs there does not exist *the* Grid. What we have learned in the last years is that there are many different Grid types satisfying different specific requirements. Following (CERN, 2007) grid systems can be categorized into the following six classes:

National Grids: The idea behind National Grids is to couple high-end resources across a nation. This will provide a strategic computing reserve and will allow substantial computing resources to be applied to large problems in times of crisis, such as to plan responses to a major environmental disaster, earthquake, or terrorist attack. Furthermore, such a

Grid will act as a national collaborator, supporting collaborative investigations of complex scientific and engineering problems, such as global climate change, space station design, and environmental cleanup. For example, the UK has a major e-Science program, part of which is dedicated to developing a major national Grid.

Private Grids (sometimes called local-Grids or intra-Grids) can be useful in many institutions (hospitals, corporations, small firms, etc). They are characterized by a relatively small scale, central management and common purpose and, in most cases, they will probably need to integrate low-cost commodity technologies. In fact, commercial solutions for such private Grids are already available, such as Entropia's DC Grid, and likely to grow in sophistication over the next years.

Project Grids will likely be created to meet the needs of a variety of multi-institutional research groups and multi-company virtual teams, to pursue short- or medium-term projects (scientific collaborations, engineering projects). A Project Grid will typically be built ad hoc from shared resources for a limited time, and focus on a specific goal. Typically, this is something a self-motivated team could set up, without need to apply to any major Public Grid Infrastructure for permission. The LHC Computing Grid (LCG) Project is an example of such a Grid for a particular experiment in high-energy physics.

Goodwill Grids are for anyone owning a computer at home who wants to donate some computer capacity to a good purpose. To date, activities in this area has been limited to the various @home projects, where the fun of participating is the main motivator, with sometimes prizes and potential fame as extra bait (imagine if your computer happened to detect the first message from outer space! SETI@home will give you a free trip to Puerto Rico if that happens, to see the radiotelescope that they are using for their project.).

Peer-to-peer Grids: Peer-to-peer technology depends on people sharing data (like the now defunct Napster and its many subsequent imitators) between their computers. The name peer-to-peer suggests that there is no central control, but actually, in the case of Napster, there was a central repository of addresses, and this is precisely how the company got nailed! However, newer versions are truly peer-to-peer, requiring no third-party intervention. Compared to Goodwill Grids, the idea here is that you get in kind for what you give: access to music files for sharing your own, for example.

Consumer Grids: In a Consumer Grid, resources are shared on a commercial basis, rather than on the basis of goodwill or mutual self-interest. Companies or other organizations rent distributed resources, and the owners of these resources are paid for the computing power or data storage capacity they contribute, by a middleman in charge of the middleware. Although many customers could use the same resources, in general they would not have common collaborative aims, and indeed there is a challenging need to provide security that prevents different customers snooping on each others' work. A big issue in such Grids will be *resource marketing*: a user has to find the resources needed to solve his particular problem, and the supplier must make potential users aware of the resources he has to offer. There has been attempts to make commercial solutions based on @home type *cycle scavenging*, but basically it is seen as generally too unreliable to be a business proposition. It has been suggested that future generations of processor chips could come with a Grid facility pre-installed, to optimize their use by Grids when not in use by the owner.

Grid Resources

This section describes the basic resources a Grid needs to have. Each node contains a certain number of computers, which may be playing different roles. To provide a concrete example, we describe here the concept of the EU DataGrid project. The DataGrid consist of approximately 1000 CPUs at 15 sites across Europe, and is built on top of the EU-funded GEANT high-speed communications network. The machines play one (or more, if possible) of the following different roles:

Resource Broker: Module that receives users' requests and queries the Information Index to find suitable resources.

Information Index: Can reside on the same machine as the Resource Broker, keeps information about the available resources.

Replica Manager: Used to coordinate file replication across the Grid from one Storage Element to another. This is useful for data redundancy but also to move data closer to the machines which will perform computation.

Replica Catalog: Can reside on the same machine as the Replica Manager, keeps information about file replicas. A logical file can be associated to one or more physical files which are replicas of the same data. Thus a logical file name can refer to one or more physical file names.

Computing Element: Module that receives job requests and delivers them to the Worker Nodes, which will perform the real work. The Computing Element provides an interface to the local batch queuing systems. A Computing Element can manage one or more Worker Nodes. A Worker Node can also be installed on the same machine as the Computing Element.

Worker Node: Machine that will process input data.

Storage Element: Machine that provides storage space to the Grid. It provides a uniform interface to different Storage Systems.

User Interface: Machine that allows users to access the Grid.

Workflow Management Systems

There are basically two different modes a Grid is used:

1. *Data oriented*: A large problem is split up into many similar sub-tasks which then are put into the Grid's job queue and computed by the worker nodes

2. *Pipeline oriented*: A large workflow consisting of many different and dependent tasks is created and for each step different jobs are submitted to the queue

According to the definition given by *Workflow Management Coalition*, a workflow is: "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" (WfMC, 2007).

A *business process* is the set of procedures required for obtaining a given result, therefore workflow is the automation of a set of operations that allows obtaining a given result, with the exchange of information among involved entities and with respect to defined procedural rules. The operations involved by a workflow are called activities; an activity is a part of the entire work and it represents a logical step in the process.

Since in the *data-oriented* case the problem can be reduced to how to split up the data the *pipeline-oriented* case is considerably more difficult. Here, the workflow must be defined, implemented, and executed. This issue of execution in a given order of complex computational tasks in a Grid environment has even been discussed by the "Grid Computing Environment Working Group", in the Global Grid Forum (Hugh P. Bivens, 2001).

To run a workflow into a Computational Grid it is necessary to define a language to be interpreted and manipulated automatically from a management system, which should allow defining a set of activities, their relation, involved entities (i.e. Applications, data resources, etc.) and some criteria for determining the start and end of the processes.

However, there are many available workflow manager that fulfill these tasks (such as Nimrod, Triana, Taverna, Pegasus, Proteus) but they need to be installed on top of an existing Grid platform and are usually not part of them (a nice exception is the myGrid project (Stevens et al., 2003)).

5.1.3 Problems of Todays Grid Systems

Nowadays most Grid systems are run by large research organizations, companies or governmental institutions such as NASA (Information Power Grid), US Department of Energy together with IBM (Science Grid) or the European Union (EGEE). These organizations have dedicated staff who set up, configure, administrate and manage these projects. This is still far away from trivial, making these administrators vital to the task. As in the case of the Internet boom in the late 1990's what is really is needed to fulfill the "everyone can use the Grid" vision is

- Significantly reduce the complexity of installing and maintaining a Grid system.
- Provide easy to use client software that can compute jobs on Grid resources.
- Allow for easy participation in a Grid system either as user who wants to compute a large problem or to provide resources - this includes things such as registration and security models.
- Merge the needed components such as the Grid platform, workflow manager and scheduler into some consistent product.
- Allow for easy enabling of Grid technology to existing algorithms.

5.2 The Quasi Ad-hoc (QAD) Grid

This section describes our quasi ad-hoc approach. Before we dive into details section 5.2.2 will give a broad overview of the basic modules. These are the building blocks of this approach. Subsequently, section 5.2.3 describes details of the underlying concepts which differ from traditional Grid solutions.

5.2.1 Why Another Grid Approach ?

- No need for real client software
- Heterogeneity: embedding of specialized hardware
- Unreliability of nodes: specialized checkpoints and migration possibilities
- Merging the grid and cluster idea
- Hot deployment of services into a running system

5.2.2 Modules

This section introduces the main *building blocks* of the quasi ad-hoc Grid from a users' perspective (see Figure 5.2.2):

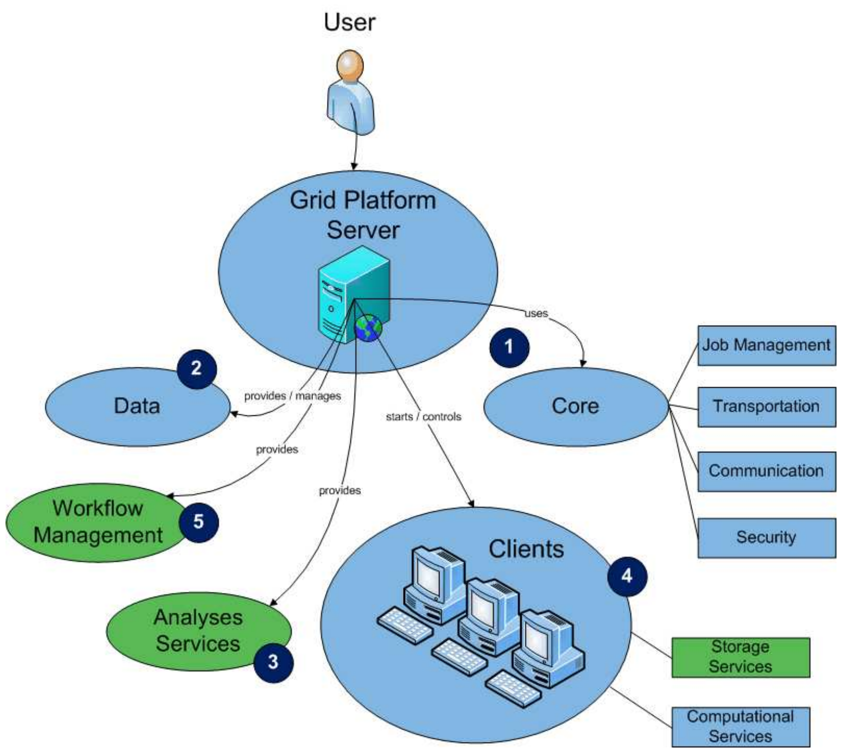


Figure 5.2.2: Grid - from the users' perspective. Elements in blue are standard Grid components; green elements are extensions.

User: The User can login to the (web-based) platform server for managing and control purposes. Depending on his credentials he might only have limited access to data, clients and Grid services / functions.

(1) Grid Platform Server: The Platform Server is the central instance of the quasi ad-hoc Grid. It allows users to use and access the Grid system e.g. via a web-based front-end (see section 6.2). The core functions are:

- Management of, enabling access to and transports of data.
- Job administration, e.g. creation and fail-over handling.
- Start and control of and communication with workers.
- Security related issues such as client login or data encryption.

To increase performance and enable fail-over handling it is possible to run many Grid Platform Servers in parallel. Using database technology these instances periodically synchronize their data, such as jobs details.

(2) Data is stored at the Grid Platform Server and replicated partly at the worker's host system. This distributes the data across the Grid and enables efficient data access for the distributed workers.

(3) Analyses Service: The server can provide complex analyses services through its web-frontend.

(4) Workers perform the computational work and mirror parts of the data. It requests jobs from the Grid's platform server, computes them and transfers the results back to the server. Additionally, it mirrors data from the platform server and provides it to other workers.

(5) Workflow Management: As described in section 5.1.2 workflows are sequentially executed tasks that depend on another. The management includes:

- Creation of workflows
- Execution of workflows. That is, after a workflow step is finished, the appropriate actions have to be taken including creation of new tasks

5.2.3 Concepts

Worker Injections: In contrast to (almost) all other Grid and cluster approaches, client software is not previously installed on target machines but on demand our platform connects to the target machine and transfers and executes the client software via SSH [4] or WMI [5] . Of course, the client can also be pre-installed and executed locally on the target machine.

[4]Secure Shell or SSH is a network protocol that allows data to be exchanged over a secure channel between two computers. Encryption provides confidentiality and integrity of data. SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

[5]The Windows Management Instrumentation (WMI) is a set of extensions to the Windows Driver Model that provides an operating system interface through which instrumented components provide information and notification. WMI is Microsoft's implementation of the Web-Based Enterprise Management (WBEM) and Common Information Model (CIM) standards from the Distributed Management Task Force (DMTF). WMI allows to manage Microsoft Windows personal computers and servers, both locally and remotely.

Job Pull: Jobs are requested (pulled) by the workers. This is opposed to most other Grid or cluster approaches where the jobs are distributed (pushed) by some service node directly to a worker.

Database Centered Communication: Current database technology is fully platform independent. In the QAD Grid the entire communication between Grid nodes (that is servers and clients) is done via modifying database entries which are read and interpreted by the single entities. In most other systems communication is done via web-services [6] which causes communication overhead and requires installation of a web-service container (usually a special server component).

Workflows / -items: Jobs in our system are workflow elements (see section 5.1.2). This means all (atomic) elements can be automatically combined to complex workflows. A workflow is a directed acyclic graph (DAG).

Data Access: All data is centrally hosted on the server platform and automatic data replication is performed to the Grid workers. That is, we can perform both *Data-follows-client* and *client-follows-data* paradigms. This implies that data to be processed by the Grid are first transferred (*staged*) to a platform server and then automatically actively distributed to the clients or passively replicated over the Grid.

Security: Security is obtained via the database security features. All database connections and client logins are encrypted and transfered over a (certificate based) SSL [7] connection. Further data connections can be secured by SSL.

(Hot) service deployment: Services[8] are deployed simply by the worker registering a new service class at the platform server. From that point on jobs can be submitted to the Grid that require that particular kind of service. *Hot* in this case means that a new service can be deployed at any time and the server does not need to be restarted. Further, a new service is automatically announced throughout the Grid by database synchronization.

Job / Worker matching: One of the most important parts of a Grid system is how the workers are matched to the available jobs. In our approach this is solved by matching a service ID. Jobs are tagged with a service ID (they require a particular service to be computed) and a worker requests

[6]The W3C defines a Web service as "a software system designed to support interoperable Machine to Machine interaction over a network." Web services are frequently just Web APIs that can be accessed over a network, such as the Internet, and executed on a remote system hosting the requested services. The W3C Web service definition encompasses many different systems, but in common usage the term refers to clients and servers that communicate using XML messages that follow a particular standard (SOAP). Common in both the field and the terminology is the assumption that there is also a machine readable description of the operations supported by the server written in the Web Services Description Language (WSDL).

[7]Secure Sockets Layer (SSL) is a cryptographic protocols that provides secure communications on the Internet. It provides endpoint authentication and communications privacy over the Internet using cryptography. Typically, only the server is authenticated (i.e., its identity is ensured) while the client remains unauthenticated; this means that the end user (whether an individual or an application, such as a Web browser) can be sure with whom they are communicating.

[8]Here, service means computational services such as a particular data analysis.

jobs from the server tagged with their own service ID. The job is then distributed to this worker.

5.3 QAD Grid Platform Server

As briefly pointed out in section 5.2.2 this is the central instance in our Grid approach. We have implemented it as a web-Application where users can login to and use it online. This means the user does not need any local program to use the service. Its functions include:

1. Management
    - Authentication / Authorization
    - (Secure) client/server communication
    - Data transportation
    - User management
    - Worker management
    - Data management
    - Service management
2. Job execution
    - Worker injections
    - Workflow execution
    - Task scheduling and provision
    - Job/worker matching
    - Data provision
3. Monitoring
    - Worker monitoring
    - Offline Machine Monitoring
    - (Workflow-)task execution monitoring
    - Result check / verification

Further, it can divide large jobs into single tasks and merge returned results.

The next sections will describe each of these points.

5.3.1 Management

Authentication

Authentication is the process of verifying a subject's (e.g. person or computer program) identity. This is needed to grant or deny access to a server or a client. However, note that it is actually not possible to confirm or prove really the identity of a subject. The best what can be done is to apply one or more tests (such as querying a password) which, if passed, have been previously declared to be sufficient to proceed. To validate a subject's identity we use a *two-factor*

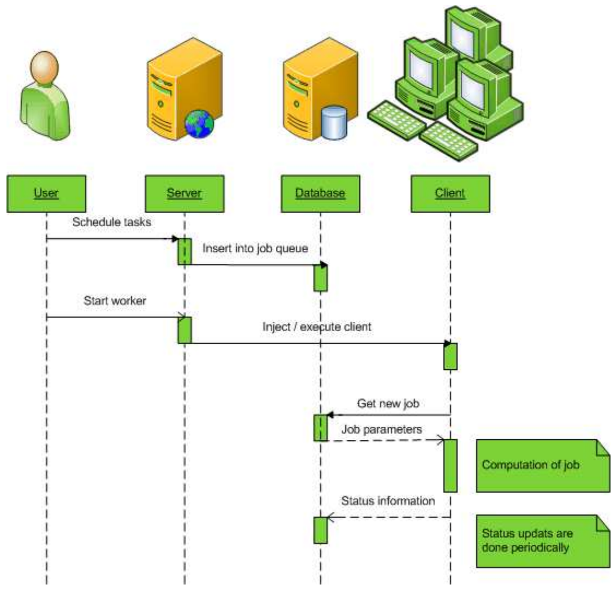


Figure 5.3.3: A user's perspective sequence diagram of the processes started when a task is scheduled or a worker started.

authentication[9] schema where two different methods are used to authenticate. Using more than one factor is supposed to be a *strong authentication* opposed to a *weak authentication* when only one factor (such as a password) is used.

Authentication in the QAD Grid is needed in three situations: (a) when a server wants to start a worker (this is operating system dependent and explained in section 5.3.3), (b) when a worker tries to connect to the Grid platform server, or (c) when a user wants to connect to the Grid server. For a worker to be able to connect to the Grid it must first register at the Grid platform server. This process is described in more details in section 5.4.2. After a successful registration process

- a database account has been created for this worker at the database server,
- the account information (user id and password) were sent to the worker,
- a new RSA public/private key pair for this worker was issued and transferred to the worker,
- a new worker certificate was issued and transferred to the worker.

Note, that transmission/communication of sensitive data is encrypted (see section 5.3.1).

The authentication process is as follows:

1. The worker sends its certificate and account information to the server. This is done through a database account that has only write access to the *connection initiation table* in the database. The worker then inserts its certificate into the table by calling a stored procedure [10] . Further

[9] An authentication factor is a piece of information.

[10] A stored procedure is a subroutine available to applications accessing a relational database system. Stored procedures are actually stored in the database. Typical uses for

information such as time stamp and the sender's IP address are stored automatically as well.

2. The server validates the certificate and checks if it belongs to the account data given. If this is the case the actual database account for this worker is activated.

3. The worker establishes a connection to the database server using its account information.

The worker is now logged into the Grid and can request and compute tasks.

Authorization

Authorization is the process of verifying that a known subject (after successful authentication) has the authority to perform a certain operation, such as accessing a particular dataset. In the QAD Grid we face the following situations where authorization is necessary:

Worker requests data: This occurs when a worker requests data belonging to the current job it computes.

Worker submits data: After computations have finished a worker submits the data back into the Grid.

User wants to use platform functions: This happens when a user is logged into the Grid platform server web front-end and requests some functionality such as job submission or data visualization.

User wants to access data: This usually takes place when a platform user wants to see raw data or computational results.

In QAD Grid we use the well-known *access control list* (ACL) approach to authorize access (see e.g. (Koch et al., 2005) and references therein). An ACL is a list of permissions attached to an object (e.g. a dataset or a function). This list specifies if a user or a worker is allowed to access this particular object and what operations are allowed to be performed on the object. A typical ACL list entry for a particular dataset could be: (user::ILM, read). This would grant user "ILM" read access to this dataset. In an ACL-based security model, each time a subject requests an object or wants to perform an operation on an object, the server first checks the object's ACL for an applicable entry and then decides whether to grant or deny this request.

(Secure) Client/Server Communication

Current database technology is fully platform independent with respect to data exchange and has implemented sophisticated security features. In the QAD Grid the entire communication is done via modifying database entries. For example, if a worker requests a new job this is done by calling a (database) method on the platform server that marks a job as in progress and passes the

stored procedures include data validation (integrated into the database) or access control mechanisms. Furthermore, stored procedures are used to consolidate and centralize logic that was originally implemented in applications. Large or complex processing that might require the execution of several SQL statements is moved into stored procedures and all applications call the procedures only.

job's parameters to the calling worker. The beauty of this is that current database technology can be used to replicate and synchronize many servers in the Grid automatically.

The main task usually performed with(-in) a Grid is computation or data analysis. Mostly this data is non-confidential but sometimes data of value are to be analyzed which need to be secured. Since usually data is replicated throughout the Grid confidential data is excluded from this process and is only provided by the server on demand. Other data needed to be securely transmitted through the Grid are e.g. account information (including passwords).

To enable secure transmission of sensitive information across the Grid we use a secure software layer (SSL) that transparently encrypts all data. This is implemented by using public-key cryptography, namely the RSA algorithm[11]. We encrypt socket communication with the (public) receiver certificate.

Communication with the database is done through an ODBC (Open Database Connectivity) driver (e.g. JDBC by Sun which is actually a bridge but fulfills the purpose). Our database is configured to only allow (SSL) encrypted connections.

Data Transportation

Data transportation between two machines is usually performed by using FTP (File Transfer Protocol), its secure S-FTP version or its Grid version Grid-FTP (Allcock et al., September 2002). The benefit of using FTP is that it is relatively easy to use, has been around for a long time and is therefore likely to be installed virtually everywhere. However, as e.g. (Huang and Grimshaw, 2006) point out, the disadvantages of FTP are numerous. First, the user must have access to an FTP account (user name/password) on the target machine. Having such access means that a user could potentially do more than just file transfer, e.g. log into the target machine and access files, directories and other machines to which he has not been given explicit access. Further, if sensitive data needs to be transfered (see section 5.3.1) the secure FTP version (SFTP) has a very big overhead. As Figure 5.3.4 shows the time needed to transport a 1GB file almost triples.

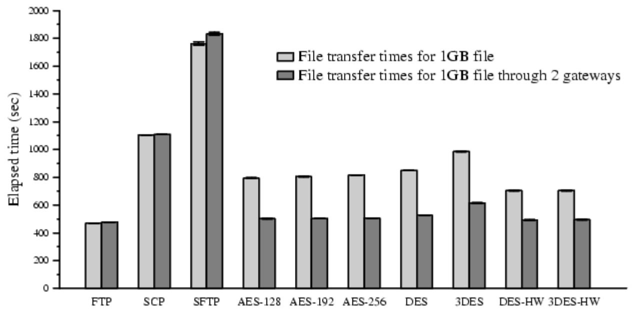


Figure 5.3.4: Benchmark values for transportation data using different encryption algorithms in comparison to plain FTP. Shown is the time (in seconds) needed to transfer a 1GB file over a 1Gbit network.

[11]Invented by R. Rivest, A. Shamir and L. Adleman at MIT in 1977, see (Rivest et al., 1978). (RSA are the initials of their surnames.)

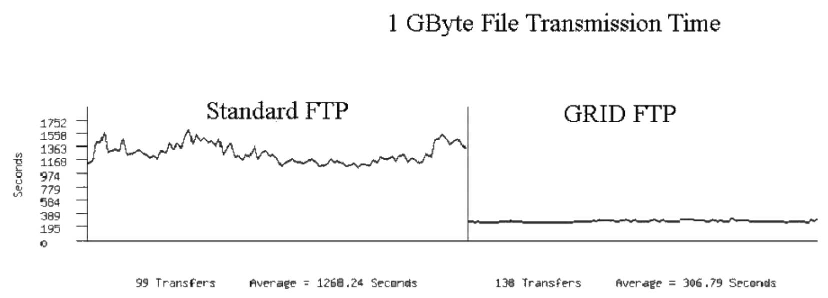


Figure 5.3.5: Comparing basic FTP to GridFTP. Shown is the time needed for the transfer of a 1GB file over the internet at different times of a day.

GridFTP solves most of these problems and extends the FTP approach by many useful features. E.g., by using parallel connections it is able to speed up transmission throughput as shown in Figure 5.3.5. But in order to use it all machines have to install the GlobusToolkit - which is definitively not desirable (see section 5.1.3).

To circumvent these problems we have developed an Internet-socked based client/server approach that allows to exchange files between two machines within the QAD Grid obeying the authentication and authorization schemas described above. A socket is one software endpoint of a bidirectional communication link between two programs in a network. Typically this would be a server program and one or more client programs communicating via a dedicated port (channel). In the QAD Grid each platform server runs at least one *data service* that accepts client connections and sends or receives data.

For a worker to get a file the following process must be successfully finished:

1. The worker queries the platform server by a database query to get IP addresses of available data services hosting the requested file (identified by the unique file ID) ordered by their local machine load level.

2. The worker measures the round-trip time (RTT) of the first five IP addresses of this list. The RTT is the time a data packet needs to travel from the worker to the service and back.

3. The worker establishes a secure socket connection (see section *Secure Communication* below) to the data service with the smallest RTT value and requests the file.

4. The data service checks the authentication and authorization of the worker and rejects the query if one fails. Authentication is done in the following way: the worker sends its account information (user id and password) and certificate to the data service. The data service logs into the platform server's database using the worker's credentials and requests the certificate of this worker. Authorization is successful if the database login is successful and both certificate are identical.

5. If the previous step succeeds the file is send to the worker. If the file is not marked as *sensitive* the connection is no longer encrypted to avoid encryption overhead. If sensitive data is to be send, first a 256bit key is sent to the worker and then encryption is changed from RSA (asymmetric) to Rijndael AES-256 (symmetric Advanced Encryption Standard, see

(Daemen and Rijmen, 1999, 2002)) encryption which is about 1000fold faster with respect to encryption and decryption.

The latter step - the fast transfer of large volumes of data - mainly relies on the network transport protocol used. However, the dominant network transport protocol of today, TCP (Transmission Control Protocol), is tuned for yesterdays wide area networks (WAN). The slow response of TCP in fast long distance networks leaves sizeable unused bandwidth in such networks. As an example, if one copies a big file (bigger than 100MB say) between two computers in a country-wide 100Mbit network the transfer rate would be about 5Mbit/s (Megabyte per second, meaning that about 95% of the bandwidth are unused.

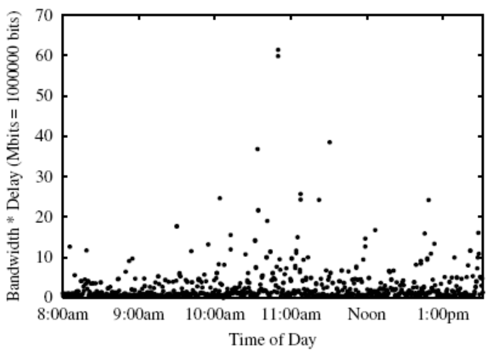


Figure 5.3.6: This graph shows how the quality of a particular connection changes between 8am and 2pm. The y-axis shows "Bandwidth · delay", sampled every 20 seconds. (Taken from (Thulasidasan et al., 2003).)

To understand this, one has to know that TCP sends data in small packages. At the start of a connection, TCP determines the size of a so-called *congestion window* to determine how many packets are send at one time. The maximum congestion window is related to the amount of buffer space that the OS allocates for the connection. To achieve maximum throughput, it is critical to use an optimal buffer size for each individual connection. Obviously, the larger the size of this congestion window, the higher the throughput. However, if set too large, the sender can overrun the receiver, which will cause packets drop on the receiver side which have to be re-sent.

For the following example to the calculation of optimal TCP buffer size we assume that there is no network congestion (no packet loss) and the connection quality stays the same over connection life-time. Then, network throughput is directly related to TCP buffer size (BS) and the network latency (NL), which is the amount of time for a packet to traverse the network.

Let us further assume that network latency in a typical connection between two computers over a 100Mbit WAN in, say, Berlin is about 40ms. We know that Windows XP has a default TCP buffer size of 17.520 bytes. The maximum possible throughput (TP) is calculated by

$$TP = \frac{BS}{NL} = 17520 bytes/0.04 sec = 0.44 \frac{MByte}{sec} = 3.5 \frac{Mbit}{sec}$$

If you would increase the TCP buffer size to 65KByte (as e.g. Mac OS does) it would get a bit better but still not close to the optimal value of 100Mbit:

$$TP = \frac{BS}{NL} = 65936 bytes/0.04 sec = 1.6 \frac{MByte}{sec} = 13 \frac{Mbit}{sec}$$

It is a rule of thumb that a near optimal TCP buffer size (BS_{opt}) for a particular connection is double the value for delay (D) times bandwidth (BW). By using a program like "ping" one can measure the round trip time (RTT) a data package needs to travel to the destination computer. The RTT is twice the delay, so we have

$$BS_{opt} = 2 \cdot D \cdot BW = RTT \cdot BW$$

If we assume a RTT of 80ms for the above connection this means that the TCP buffer should be

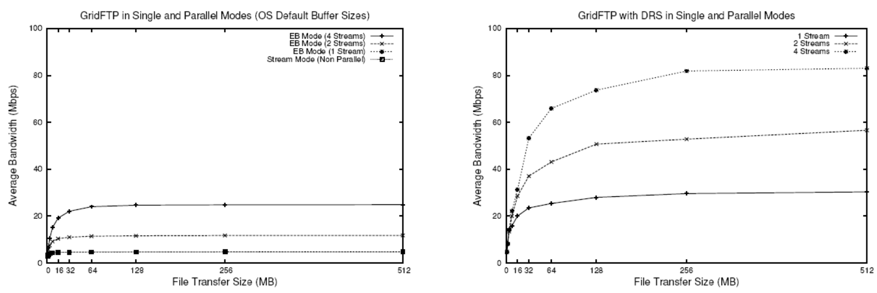


Figure 5.3.7: Performance of GridFTP with OS-default buffer size (64KB, left) versus Performance of GridFTP with dynamically adapted buffer sizes (right). Shown are the average bandwidth values depending on the size of the transferred file (16, 32, 64, 128, 256, 512 MB).

$$BS_{opt} = 0.08sec \cdot 100\frac{Mbit}{sec} = 8\frac{Mbit}{sec}$$

Since with a transmission rate of 1 Mbit/Sec one can transfer 0.125 MByte/Sec we get:

$$BS_{opt} = 8\frac{Mbit}{sec}/8 = 1MByte$$

Therefore, the TCP buffer size for this connection should be set to 1 MByte.

Now, in reality the assumption made at the beginning are not very likely to hold. First, if packet loss starts to occur (Mathis et al., 1997) have shown that throughput is then bounded by the Maximum Segment size (MSS), which is Maximum Transmission Unit[12] (MTU) minus TCP/IP headers (in practice default values are: MTU: 1500bytes, TCP/IP Header: 40bytes and therefore MSS=1460bytes).

$$Throughput \propto\sim 0.7 \cdot \frac{MSS}{RTT \cdot \sqrt{packet\_loss}}$$

This problem can be tackled by increasing the frame size to about 8KByte (Chase et al., 2001), that is using so-called *Jumbo Frames.* Since the maximum frame size is set by switches in the Internet on the path from sender to receiver we cannot do anything about this. The second problem we face is that the bandwidth delay (RTT values) can fluctuate quite wildly over lifetime as shown in Figure 5.3.6. Following (Thulasidasan et al., 2003) for the transmission part we have implemented dynamically adaptive TCP buffer sizes to allow TCP flow control to adapt to changing high-speed WAN environments, especially when transmitting large files. Figure 5.3.7 shows the effect of this tuning in a concrete example where this technique was used in GridFTP. It can be easily seen that performance approximately quadruplicates.

User Management

In the QAD Grid a user is modeled as a database object with the following properties:

Account information: This can be a local account or a network account. In both cases a user name is stored. In the local case an additional

[12]Size of the largest packet that a network protocol can transmit.

password (actually its MD5 hash value) is stored as well, which is used for authentication. In the network case authentication is done by querying a network authentication server, e.g. via LDAP or Kerberos protocol.

Group membership: A users can be member of one or many groups (e.g. an institution) which can be used to give many users at once access to particular objects, such as data or functions. Using this feature it is also possible to implement some kind of hierarchy, e.g. simple user or administrator groups.

Account details: Further (optional) information such as real name, institutional affiliation, e-mail address and so forth can also be stored.

Billing details: Since system usage logs are collected automatically (such as CPU time used for computation) these information can be used to implement some kind of billing.

To be able to login to the QAD Grid a user needs to be registered at the platform server. This is done through a special web-site new users can access. At this site they enter their user details which then needs to be reviewed by an administrator. After successful registration a user can log into the web-based front-end of the platform server and use its services, such as start of new computations or analyses (see section 5.5) or view results of previous runs. After successful login a fine-grained access control list (ACL) system (see section 5.3.1) is used to determine (1) what parts of the system the user can see, what (2) functions he can use and (3) what data and results he is allowed to see.

(1) Access to web-sites: When a user accesses a web-page the web-server checks during the *on-load sequence* of this site whether there exists an ACL entry that allows this user to see this site.

(2) Use of functions: As in the web-site case each time a user requests the use of a function, e.g. start of a computation, the web-server first checks if the users has appropriate credentials to use this.

(3) Access to data: There are many scenarios when a user needs to access data, e.g. visualization of results, computations that need data or just display of raw data. Again, there exist ACL entries for each dataset available in the system that is checked if data is requested either through the web-server or directly from the database.

Worker Management

A worker is modeled as a database object with the following properties:

Account information: This includes a user name and a password and is mainly used to log into the database.

Public key: The public key is used by the platform server to encrypt communication (initialized by the server) with this worker.

Certificate: The worker-specific certificate is issued the first time a worker successfully registers at the platform server. It contains a MD5 digest of a combination of the service id this worker offers and the public key of this worker. It is used for authentication purposes.

Type of service offered: This service id is a string describing the (computational) service this worker offers. In the job/worker match process (see section 5.3.3) this is used to match workers with jobs.

Worker machine details: This comprises details about the machine the worker runs on: information about CPU speed, memory size, operating system and maximum size of storage the worker is allowed to use for temporary files.

Connection details: Here information about current and previous connections are stored, covering connection speed (RTT and up-/download speed), development of connection speed (only RTT), presumed (physical) location and timestamps of connection initiation and termination.

As briefly described in section 5.3.1 a worker needs to be registered at the QAD Grid platform server to be able to login to the Grid and offer computational services (for details see section 5.4.2). If a worker is already registered at the Grid it just has to go through the authentication procedure. During the life-time of this connection many information are gathered and stored at the platform server. After successful login a worker's fingerprint is created that contains the following information:

- timestamp of connection start
- connection speed, that is RTT and time needed for upload and download of a 500KB file (upload is measured by uploading 500KB binary content to the platform server by FTP)
- (presumed) geographical location (city, country, latitude and longitude)[13]

Further, the worker sends a list of files (file IDs and MD5 checksums) currently available locally at the machine it was started on.

To keep this information up-to-date a worker sends every 5-300 seconds an "I-am-alive" message to the server that updates the information about state, local load and current connection quality (see section 5.4.3 for details). Using this information conclusions can be drawn about the overall connection quality over the connection life time. Further, we can infer indications about how to mirror data cross the Grid smartly in order to minimize time needed for a worker to transfer needed data (see section 5.3.2).

Data Management

To model and manage data in the QAD Grid we use a hierarchical study-centered approach. That is, all data stored or used (e.g. analyzed) in the Grid is at least linked to a study (e.g. "Study of Leipzig Blood-Donators in 2002"). Further hierarchy levels are:

Data or result groups (e.g. "Men above 20", "Results of Peak Picking Analysis 20" or "all 1D spectra of a 2D spectrum map"): Often, there are many raw data files or many results entries that actually belong together and form a dataset. For example, a raw 2D spectrum consists of many 1D data files.

[13]Using a webservice such as `http://hostip.info/`. These services are of course not always absolutely correct but it can be used as a good estimate.

Data elements (e.g. a single 1D spectrum, or a peak picking result). This is the most basic element reflecting the actual information physically stored in the Grid. There exist two classes of data:

1. RAW Data: This kind of data usually exists as text or binary files on the platform server. These files can get quite large and easily exceed several gigabytes per file. Depending on the network path transferring these files can take several minutes to several hours. Especially if large quantities of data is requested from a single node network traffic speed can decrease drastically.
2. Analysis (intermediate) Result Data: This type of data is usually stored in database tables and fetched by database queries. The size of data fetched per query is usually comparatively low. However, if many workers query a database the overall performance drops as in the file based case.

To model this we have chosen to use a rooted directed acyclic graph (DAG). This naturally introduces hierarchies and links between elements and allows for fast searches. Studies are directly connected to the root, groups and datasets are part of a study and data elements are modeled as the leafs. Each leaf in a tree corresponds to an actual physical file or database entry of a result.

A nice feature of this structure is that is allows for linking any kind of information to each element (node) without losing the DAG's properties. For example, metadata (e.g. patient metadata, timestamp when a datum was added to the system, or information about physical location of a file) can be directly linked to a leaf. Especially the latter will be of importance for data distribution and (subsequent) data location in the Grid (see section 5.3.2).

Task Management

To split up a large problem some problem specific method must break this up into smaller tasks. These tasks contain parameters describing

- system data, such as: task priority, task owner, draft flag and linked task (see section 5.3.4)
- target specification: sometimes a particular worker has to handle a task
- the type of this task (e.g. peak picking or file copy)
- (optional) dependencies on another tasks
- (optional) target worker that has to handle this particular task
- what input data is needed
- where results should be written to (e.g. database or file)
- further task dependent parameters - such as fitting variables or window sizes

5.3.2 Providing Data in the Grid

As described in the previous section the QAD Grid provides two different types of data: (a) files (usually RAW data) and (b) database entries (normally meta data or analysis results).

In a single-server based setting with many workers just transferring data to the workers would cause a very high CPU load. Whereas the latter case is solved by using many (automatically) synchronized[14] database server the former case is more complicated. To avoid high network traffic on the platform server and to enable workers getting their requested data from fast (near-by) sources we developed a peer-to-peer approach to distribute (mirror) data cross the Grid. This is done automatically and on on special events (see below).

The basic transport mechanisms are described in section 5.3.1. The following paragraphs give details about the distribution algorithm. The key ideas are as follows:

- All data is available from the platform server (master repository)
- Workers can be assigned storage space on their local host
- Selected data is mirrored at reliable workers with good connections based on their geographical location
- Data is copied automatically when server and workers are idle by creating a task that states a particular worker to handle this task
- For each file successfully copied to a worker an entry (MD5 checksum) in the platform server's data graph (see section 5.3.1) is created that states that this worker now has a copy of this file.

This system is usually called a *Data Grid Management System* (DGMS). The next sections describe how data is selected that is going to be mirrored and how the workers are selected that this data is copied to.

Selecting Workers to Mirror Data

This step selects the workers to be used to mirror data within the Grid. The main target is to find some *hubs* that are then used to distribute data from the main server(s) into several geographical areas where many computing workers need data. We therefore (a) decrease load on the central servers caused by data transmission and (b) use short distance network connections and save bandwidth. We have selected to use a maximum of 10% of all workers being online at a given time as mirror nodes, since tests have shown this seems to be the minumum of nodes necessary to provide the data without creating bottlenecks. These workers need to meet two criteria:

Reliability: The worker must have been online on average at least one hour during the last five times they went online.

Speed: The upload speed of a worker must be higher than the bottom 80% out of all workers being online at the time the measurement is taken.

[14]E.g. using Microsoft's database mirroring features.

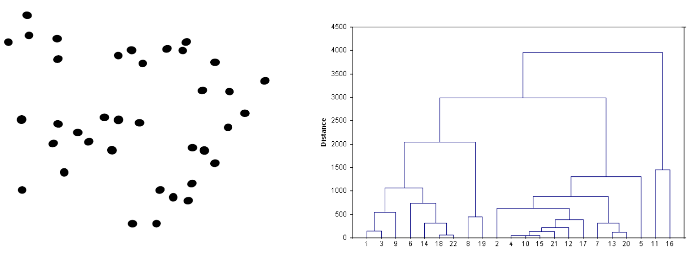


Figure 5.3.8: Example for a hierarchical clustering (HC): the nodes shown on the left are clustered by HC (single linkage) using Euclidean distances. The resulting dendrogram (right) shows the results.

Out of this list of all workers being *reliable* and *fast enough* a distance matrix is created. These geographical distance between two workers is calculated using the haversine formula (Sinnott, 1984). This computes the great-circle distances between two points on a sphere given their longitudes and latitudes which is particularly well-conditioned even at very small distances. On this distance matrix a (single linkage) hierarchical clustering (Johnson, 1967) is performed. Searching from the top in the resulting dendrogram (see Figure 5.8(b)) that level is sought for that maximizes the number of clusters but has at most c clusters. c was set a-priori to 10% of the number of workers. What also could have been done is to use a technique known as multi dimensional scaling (MDS) (Shepard, 1962) to recover the original Euclidean coordinates and find clusters on the resulting map.

Within each clusters found the most reliable and fastest node is then selected for mirroring.

Selecting Data to be Mirrored

Not all data in the Grid system is used all the time: if analyses have been finished on a particular dataset it might never be used again. On the other hand a particular (presumably) quite recent dataset will be analyzed on many machines at the same time and long delays can occur if data is copied from a single source. Therefore, it does not make sense to mirror all datasets across the Grid but for some data it is extremely interesting to make it highly-available during load spikes. Thus, the system has to select datasets that are to be distributed (mirrored) across the Grid. The actual selection happens in *rounds*, that is, each time the distribution process is started datasets of a maximum of 5GB are selected. The data selection algorithm is organized in two stages and works as follows:

Server Stage: At this stage data is selected from the central server to be copied to the workers using the *last in, first out* principle. This means, first all datasets are selected that are not currently distributed within the Grid more than three times. The resulting list is then sorted by time and date they were added to the system. Starting from the most recent item datasets are selected up to a (total) maximum filesize of 5GByte.

P2P Stage: At this stage data is distributed directly between workers. The procedure is similar to the server stage, but this time five copies of the most recent datasets are created and distributed to the workers. Tests have shown that five seems to be the minimum number to avoid bottlenecks in our testbed.

To distribute data the system creates tasks (see section 5.3.1) that are inserted into the system's job queue with an assigned priority that is lower than analysis tasks. Therefore, if a new analysis is queued it will be handled before the next copy task will be executed.

Remarks

To disable this feature set worker storage size to zero. This might be useful in local area networks where data is available through the network file system anyway and does not need to be copied.

5.3.3 Job Execution

In this section we describe how jobs are actually created and executed. This comprises starting of workers at target machines (that will do the actual computational work), creation and scheduling of tasks and the actual execution of tasks, that is matching of tasks and workers. Further, we give details about how workers get job specific data from the Grid.

Worker Injections

One of the most striking features of the QAD Grid approach is that there is no need to (pre-)install client software on the working machines. The only requirement is that the target machine provides SSH access. When a machine is added to the Grid the platform server logs into that machine, copies the client software and needed libraries to this target and executes it. The worker then connects itself to the Grid and starts working (see section 5.4). We have tested this with different types of Linux, MacOS and Windows systems running a variety of SSH servers. If a target machine cannot provide SSH access the worker can of course be started manually on that machine. Another possibility on Windows-based systems is the usage of WMI (Windows Management Instrumentation) which is also supported. This allows the QAD Grid Server to log into an windows machine and - as in the SSH case - remotely execute a program.

The main advantage of this approach is that only the network (IP) address of a new client needs to be added to the QAD Grids client database and there must exist a user account on that machine for the QAD Grid server to login. Then, the server can inject the client software and start a worker on that machine without further user interaction. These workers are then fully controlled by the QAD platform server.

Details of this injection process are as follows:

1. The QAD Grid server establishes a SSH (Linux) or WMD (Windows) connection to the target machine.

2. If no sub-directory "qad_grid" exists within the clients temporary directory, it is created.

3. The "qad_grid/3rd_party_libraries" sub-directory is checked whether all libraries needed by this worker are available. If not, missing libraries are transferred via SCP.

4. The "qad_grid/workers" sub-directory is checked for the existence of the worker to start. If it does not exists or the available version is outdated

the current version of the worker is transferred as a zip archive and unzipped into a new sub-directory.

5. Further commands - as stored in the database for this kind of worker - are executed.

6. The worker is started via the worker specific command line stored in the QAD Grid's database during *registration* (see section 5.5.2.

Task Scheduling and Provision

One of the central functions of the Grid platform server is to provide jobs and their respective details for workers. This means, there must exist two basic functions on the platform server:

- Receive jobs from some instance and insert them into the central job queue. The jobs can be send from some (authorized) worker or from a Grid platform itself where a user has started an analysis that results in a set of jobs.

- Provide jobs to workers: each authorized worker can request jobs of a particular kind from the server (see below). A job contains all needed information the workers needs, such as algorithm parameters, location of data to be analyzed etc.

Job/Worker Matching

Each worker can handle exactly one particular kind of job, such as copy a file, perform an analysis or classify an item (see section 5.4). Hence, each job and each worker is assigned a so called *job type id* (JTI) tag. If a worker requests a job it sends its JTI tag and the platform server checks if unprocessed jobs tagged with this JTI exist. If this is the case the first job in the queue is marked "in progress" and the parameters transferred to the requesting worker.

Requesting Data

To handle a task a worker mostly needs a dataset to e.g. perform an analysis on. This data is stored at the central Grid server and usually at some workers within the Grid. To get this data the worker queries the platform server to get a list of all nodes that currently host that particular dataset. The request includes the geographical location and the id of the needed dataset. The resulting list includes the machine's IP addresses ordered by the (geographical) distance to the requesting worker. We could also have used the upload speed of the target as order criterion but as hosting nodes must have a large upload bandwidth the geographical location is considered to be more important to save total network (Internet) bandwidth.

Using the resulting list the worker tries to connect to and request from the closest node to get the needed data (see section 5.3.1). If a connection fails it will try the next machine. If all connections fail it will request the data from the central server.

5.3.4 Monitoring

One of the central features of the QAD Grid platform are the monitoring functions and reactions on events such as worker failure. The next sections describe what is monitored in the system, what events can occur and what actions are taken if particular events occur.

Worker Monitoring

The worker monitoring comprises two things:

Alive check: if a worker does not send a status message at least every 300 seconds to the QAD platform server (see section 5.4.3) a worker is considered lost. In this case the platform server will update the database and mark this node as down. If the node that was lost was controlled by the QAD Grid server the platform will try to re-connect to the client machine and restart the worker (client software) as described in section 5.3.3.

Workload check: If the local load of a worker exceeds a certain threshold the worker is considered overloaded (see section 5.4.3). If so, this worker is set into paused state in which a worker will not request new tasks and sleep until the workload gets again to a level below the threshold. If the worker computes a task at the moment it was set into the sleep state it will mark this task as suspended and create a dump file of the current task state. This file contains all variables necessary to restart this task on another worker / machine at exactly this position (see section 5.4.4). This file is then copied to the QAD platform server which will distribute it to another client machine that will resume from this state.

Offline Machine Monitoring

In addition to active workers the system can also monitor client machines that are registered at the QAD Grid system as possible computing nodes and allow the QAD Grid server to login but are not running a worker. This is done by periodically logging into the target machine and running a quick status check. The resulting values and machine details such as CPU speed and available memory are then inserted into the Grid server's database.

Task Execution Monitoring

This monitoring checks whether the execution of a tasks was interrupted or terminated because of a worker failure.

The former condition can fail if a node is lost during task execution (see previous section). In this case the state of this task (at the QAD's platform server) will be set to new, that is, another worker can request and compute this task.

Result Check/Verification

In some cases an explicit result verification is necessary. To achieve this a task will be computed by two different workers and the results compared prior to insertion into the database. This is done by setting the draft flag of the two (identical) task and inserting the ID of the opposite task into the linked task field. A worker that computes a task with a draft flag will not directly insert the result into the database or copy into a target repository but will send it into a *to be verified zone* on the QAD platform server. If the results from the different workers have arrived the platform server compares and either stores or rejects them. In the latter case two new tasks are created until the two results are identical.

5.4 QAD Grid Worker

The QAD Grid worker is a program that runs on a client machine and can perform exactly one particular kind of analysis / computation (such as peak picking) or service (e.g. convert a file). As described in section 5.3.1 communication within the Grid (that is between servers and workers) is entirely done via database entries. The most striking feature of a QAD Grid worker is that

- it can be written in any programming language
- runs on (almost) any hardware / operating system platform
- can contain arbitrarily complex algorithms
- can perform system calls (e.g. OS dependent copy functions)
- can enable non-Grid enabled executable programs (binaries or scripts)

A worker can be tailored to specific needs and written in any programming language that supports (T-)SQL database access. The following sections describe the basic functions a worker needs to support and an extended standard reference implementation. Further, details of worker integration into the QAD Grid are given.

5.4.1 Functionality

Base

As described above a worker essentially runs on some client machine and computes available jobs. Therefore, a worker needs to have these base features:

- Connect to and register at the Grid's Platform Server. Registration includes the announcement of what kind of job it can compute (see section 5.3.1).
- Request a job and respective parameters to compute
- Load needed data (see section 5.3.2).
- Compute job
- Transfer results back into the Grid (see section 5.3.1).
- Send alive and status messages (e.g. local workload) to the Grid

Extended

Additionally to the base methods the standard reference worker has some extended features that allow for a more sophisticated usage. This includes:

- Copy and mirror data from the Platform Server and provide these data to other clients (see section 5.3.2).
- Store intermediate results / checkpoints for possible resume of job (e.g. after node has crashed)
- Migrate to other Grid nodes (e.g. if local load becomes too high)

Universality

As briefly mentioned above, a worker can also be a simple proxy service. This means it communicates with the QAD Grid to request tasks and their respective parameters and then starts an external program with these parameters. After the external program has finished the worker handles and presumably transfers the results into the QAD Grid.

Design Principles

Each worker type has is derived from the *Base Worker* that offers the following functionality:

Communication with the QAD Grid: Registering, getting and setting status information

Data transfer: Each worker can exchange data by *File Transfer Protocol* (FTP), *Secure Copy* (SCP - copying using the Secure Shell), *copy* using network shares.

Basic functions: Log in/out, getting new jobs, migration

Based on this, the following worker categories can be defined:

Wrapper: Calling an external tool

Proxy: Using external libraries

Full implementation: Implementing a full algorithm within the worker

5.4.2 Registration

If a worker connects to the Grid for the first time it must pass through the *first time registration process*. Once registered it can use the credentials gathered during this procedure at later logins for authentication (see section 5.3.1). This process works as follows - recall that all communication via database is automatically encrypted via SSL:

1. Using a database guest account the worker inserts an *account request* into a special table by calling a stored procedure with two parameters: a worker generated 2048bit key and the id of the service this worker offers. Both values are encrypted with the public server key. The server returns an request id.

2. This request triggers another stored procedure that performs the following steps:

   - If there exists a request within the last five minutes containing the same key or the same IP address the whole request is canceled. This is done to prevent abusive generation of worker accounts which might result in high load on the server.
   - A new database account is created with randomly generated user name and password.
   - A new public/private key pair for asymmetric RSA encryption is generated.
   - A new certificate is created that is based on the MD5 digest of a combination of the requesting worker's service id and the just generated public key for this worker.
   - A new table entry is created that contains these information (DB account, key pair, certificate) and is encrypted by the 2048bit key the worker sent at initialization as binary array. This entry is identified by the request id the server sent back to the worker.

3. The worker fetches this information by database query, decrypts it and splits it back to the single components. These are then stored at the worker's host.

4. A new worker entry is created into the worker table of the platform server.

5. The worker is now registered and can log into the QAD Grid system with these credentials.

At later logins the worker simply authenticates using these credentials and announces itself at the Grid. This announcement includes a unique identification key that is created when a worker starts up and used to identify each worker, e.g. for status updates.

5.4.3 Status Messages

Since the workers and client machines are not directly controlled by the Grid (as opposed to a cluster) the Grid needs to get messages from the clients to get information about their status. This is by updates to a status table that contains a record for each client currently registered at the Grid. These updates are sent every 5-300 seconds by the workers and contain the following informations:

State: The state a worker is in is actually a text string. This string can be set during computations and can e.g. contain progress information or state changes during algorithm execution. It is mainly intended for humans to get information about the actual worker condition.

Local load: This measures (estimates) the load of the host system the worker is running on, caused by *foreign processes*. We define load as the time needed by the operating system to give control once to each other process currently running on this machine before returning to the worker thread [15] (see below for details).

Connection quality: The connection quality is measured with respect to the RTT - the time a data packet needs to travel from the worker to the platform server and back.

Each time a status update is performed a timestamp is automatically set to this database entry. Therefore, the time of the latest update can be determined. If the status has not been sent within the last 300 seconds a client is considered lost and will be logged out.

Workload Determination

As stated above, we define the machine's local load as the time needed by the operating system to cycle through all processes currently running on this machine excluding the worker process. In our worker reference implementation we use the Java method *Thread.yield()*[16] that

> "Causes the currently executing thread object to temporarily pause and allow other threads to execute." (SUNMicrosystems, 2006)

So, if a thread executes yield, it is suspended and the CPU is given to some other runnable thread. It then sleeps until the CPU becomes available again. Technically put, the executing thread is put back into the ready queue of the processor and waits for its next turn.

This means, if the worker is the only (high priority) program (thread) on a machine the time needed for the yield will be almost zero because there is no other program that will consume time. The beauty of this approach is that the CPU utilization can be well close to 100% but if caused exclusively by the worker the local load is about zero, because the worker thread is the only program running. On the other hand, if there are many CPU intensive processes running on the host machine it will take a long time for the worker thread to get back control. This time is measured and can be directly related to the number of running other threads and their CPU consumption as can be seen in Figure 5.4.9. Further, this tool gives us only the utilization of the processor core we are really working on.

Other measure such as CPU utilization (Windows) or workload (Unix) have the disadvantage that (a) it cannot be distinguished what process causes the load (is it us or the others?) and (b) it is not entirely clear what is actually measured. For example, the *average load* in the Linux world (which is often mistakenly taken as the CPU utilization by many benchmarks) is actually an exponentially-damped moving average of the total CPU queue length (see e.g. (O'Reilly et al., 1997)).

[15]A thread in computer science is short for a thread of execution. Threads are a way for a program to fork (or split) itself into two or more simultaneously (or pseudo-simultaneously) running tasks. Threads and processes differ from one operating system to another but, in general, a thread is contained inside a process and different threads of the same process share some resources while different processes do not. Multiple threads can be executed in parallel on many computer systems. This multithreading generally occurs by time slicing (similar to time-division multiplexing), wherein a single processor switches between different threads, in which case the processing is not literally simultaneous, for the single processor is really doing only one thing at a time.

[16]The yield() method is available in almost all programming languages that provide the thread concept.

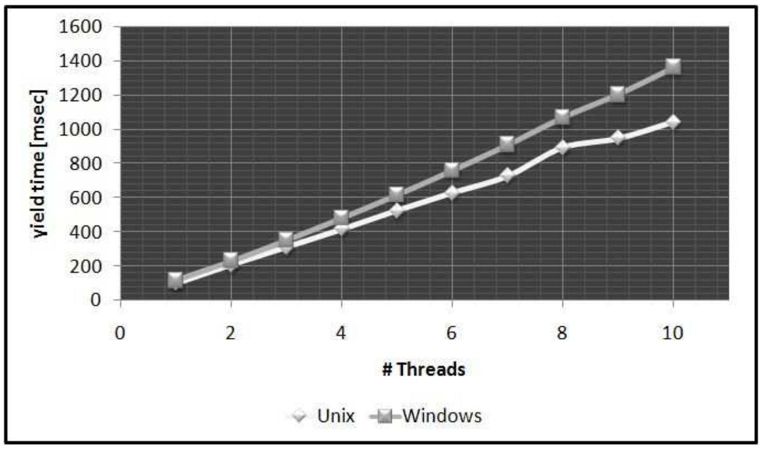


Figure 5.4.9: Time needed for the OS to cycle through the active thread (yield) vs. number of running threads. This shows that there exists an almost linear relationship. Data was created on a 2.3GHz single-core machine (2GB RAM) running Debian Linux and Windows Vista, respectively.

5.4.4 Worker Migration

So far, we presented a system that can collect and manage jobs that are then requested by workers running on some client machine performing the computation. At the moment a new job becomes available it is easy for a client machine to decide whether to request or to skip it, given that a worker has knowledge about the current system state (e.g. CPU utilization) of its host computer. This concept is known as *static scheduling*, an early approach being the Linda system (Carriero and Gelernter, 1986).

The problem now is that these conditions can (and mostly will) change over time. So using static scheduling might be a good idea within a dedicated cluster environment but as soon as user's workstations are integrated the following obvious problem arises: A user, say Alice, integrates her desktop workstation into the Grid and goes away for some time. Another User, say Bob, creates a job in the Grid system that is then requested by the worker running on Alice's machine. The trouble comes if Alice returns to her workstation before Bob's job has finished. If static scheduling is used, the options are limited: (a) we can allow Bob's job to finish which will make Alice unhappy because her system will be slow or (b) terminate Bob's job that will make Bob unhappy because he loses work already done.

In the QAD Grid system it is of primary importance that an owner of a workstation does not pay penalty for integrating its machine into the Grid, that is running a worker on his/her machine. So, workers must have the ability to vacate a workstation (or at least pause computation) if a users starts working on that machine and/or wants the worker to quit. In the remaining of this section we will describe a dynamic scheduling approach that is able to satisfy both, Alice and Bob. The key idea is that when Alice returns and starts using her machine Bob's job is paused, transferred to another workstation that is idle at that moment and continued from the point it was stopped. This concept holds also for multi-core systems where the load of one of the cores might get too high.

Worker migration (also known as Process Migration) is the process of freezing a workers state, sending that state along with the worker to another idle workstation, start that worker at that workstation initialized with the restored state and continue execution. There are many different approaches to this such as agent-based systems (Baumann et al., 1997), kernel-based (such as MOSIX or Condor (Barak and La'adan, 1998; M. J. Litzkow and Mutka, 1988)) or user-space based (e.g. Sprite (Douglis, 1990)) scheduler that support transparent migration. The most apparent downside of all of these approaches is that they require a particular operating system and programming language. This is because they are either built into the kernel or have to be linked to the executable program or need to be programmed in a specific framework.

To overcome these issues we have developed a migration concept that is realizable in most programming languages and implemented it in our worker reference implementation. Because this implementation is done in Java and does not need any hardware specific hooks it will be usable on most architecture/operating system combinations.

Migration Algorithm

Although there are many different migration strategies and implementations, most of them share a similar design and can be summarized in the following steps:

1. A migration event is created by the client, negotiated with the server and finally accepted.
2. The process to migrate is suspended and its state changed accordingly.
3. The process state is extracted and stored locally (see below).
4. The stored state is either transferred to the server (to be distributed later) or directly to a destination node.
5. A new process instance is created on a destination node and the stored state is imported.
6. The new instance is resumed. Migration is now complete and the stored state can be deleted from the source machine.

Migration Policies

In the QAD Grid we use the so-called *sender-initiated policy.* This means, that a worker is aware of its host environment and decides when this node is overloaded and the worker needs to be transferred to another node. (Eager et al., 1985) have shown that this strategy is convenient for systems with light and moderate load and better suited when compared with other concepts, such as receiver-initiated policy (underloaded nodes request processes from overloaded nodes) or symmetric policy (combination of the former two).

At special checkpoints (see below) a worker performs a *health check* of its environment. The main component here is the local workload as described in section 5.4.3. If a worker identifies an unhealthy condition, that is the local workload exceeds a predefined threshold, it sends a migration request to the platform server and pauses until it gets a reply.

Checkpoints

Checkpoints are particular breakpoints in a program defined on the source code level. At these points the execution of the worker's core algorithms is interrupted and control is given to a (implementation specific) checkpoint handler. Obviously, it is the programmer's responsibility to integrate enough checkpoints at reasonable positions. Within this checkpoint method the following actions should be performed:

1. Check the runtime of the process. Measurements have shown that it is only advantageous with respect to the total execution time ("run slow" vs. "migrate and run faster") if a process runs at least 3-5 minutes.

2. Check the current health condition of the machine hosting this worker. The key property used here is the workload of a client as described in section 5.4.3. Tests have shown that a workload above 100ms, that is, it takes 100ms for the worker process to get back control from the operating system, means the worker process runs significantly slower that normal. Therefore, we consider a worker's host system as overloaded when the workload exceeds 100ms.

3. If the health check shows no overload condition control is given back to the core algorithm.

4. If the worker is overloaded it sends a signal to the server and requests to be migrated. If the server does not answer within 10 seconds or refuses the migration control is again given back to the core algorithm.

5. If the migration request is accepted migration is performed as described above and the worker terminates itself.

With termination of the worker the checkpoint handling is finished.

Worker Persistence

To realize a worker's migration the worker must be capable of saving its execution state. This property is called persistence and is a two step procedure: (1) getting the process' state (variables, stack and the point of execution) and (2) storing the state into a file that can be transmitted over a network. Sophisticate (but operating system dependent) systems can interrupt a process at arbitrary points. To retain OS independence we cannot use this technique and require each worker to define its own checkpoints (see above).

The key design pattern in our QAD Grid workers is separation of concerns (SoC) which means to break up a computer program into distinct features that overlap in functionality as little as possible. A concern is any piece of interest or focus in a program. In our case, what needs to be implemented is the separation of data structures needed for the core algorithm (core data) and data necessary for the peripheral program. Obviously, only the core data needs to be stored and implanted into the target worker that is going to continue from the point the source worker has suspended. Consequently, for the worker to store (and eventually load) its core data each data element needs

a) to be registered in and accessible through a globally (but within this particular worker instance) available list,

b) to have a method to dump its content to a given stream object [17] and

c) to have a method to load its content from a given stream.

A list element consists of the variable name and a link to the object it represents. When a worker then calls its *save state method* the list is traversed and for each object in this list the specific save method is called and the content is appended to the stream.

The above definition is quite general and needs to be specified further to allow for the key issue here, namely to enable storing and reading data in a system independent way. What we really need is a mechanism that allows for, say, storing the state of a worker written in Java running on a Linux system, and restoring this state on a Windows box, running a C++ worker. There are many commercial and open-source solutions available for this problem, for example CORBA, RPC, (D)COM or SOAP (for an overview see (Emmerich and Kaveh, 2002; Elfwing et al., 2002) and references therein). They all have in common that they use some interface definition language (IDL) to describe a software component's interface and are also capable to transfer values (and mapping types) between systems. The main reason why we developed our own (proprietary) approach here is that neither of the systems really works in practice if used with more than a couple of programming languages, according to our experience. This mainly relies on the fact that all tested open source implementations of these standards were incomplete or inadequate (see e.g. (Henning, 2006) and references therein) and generated APIs that are incoherent, strange or even impossible to use.

To address this, we are using a combination of enterprise Application patterns called *Domain Model* [18] and *Active Record* [19] (see eg (Fowler et al., 2003)). This means, each object (such as a peak, a spectrum or a peak assignment result) used in an algorithm can be mapped to a database object and hence stored in a database. The beauty of this approach is that it is fully programming language and OS independent, while allowing to simply reference large objects (such as 2D spectra) rather than copying them. For example, if a peak picking algorithm is analyzing a 2GByte 2D spectrum and now is about to store its state it does not need to store the 2D spectrum to the

[17] A stream is a source or sink of data, usually individual bytes or characters. Streams are an abstraction used when reading or writing files, or communicating over network sockets.

[18] A domain model can be thought of as a conceptual model of a system which describes the various entities involved in that system and their relationships. The domain model is created to document the key concepts and the vocabulary of the system. The model displays the relationships among all major entities within the system and usually identifies their important methods and attributes. This means that the model provides a structural view of the system which is normally complemented by the dynamic views in Use Case models. An important benefit of a domain model is to describe and constrain system scope. The domain model can be used at a low level in the software development cycle since the semantics shown therein can be used in the source code. Entities become classes, while methods and attributes can be carried directly to the source code; the same names typically appear in the source code.

[19] Active record is an approach to accessing data in a database. A database table or view is wrapped into a class, thus an object instance is tied to a single row in the table. After creation of an object, a new row is added to the table upon save. Any object loaded gets its information from the database; when an object is updated, the corresponding row in the table is also updated. The wrapper class implements accessor methods or properties for each column in the table or view. This pattern is commonly used by object persistence tools, and in object-relational mapping. Typically foreign key relationships will be exposed as an object instance of the appropriate type via a property. Implementations of Active Record can be found in various frameworks for many programming environments.

database but simply inserts the ID of that 2D spectrum since it is available at the database or at the central server anyway. Therefore, only references to existing objects and real dynamic data needs to be stored. The actual storing process works as follows:

1. In the *migration table* a new record is inserted that consists of the worker ID and the current migration status, beginning with "migration started".

2. The *core data list* (see above) is traversed and for each object the *store method* is called.

3. Within the store method a database entry in the *migration data table* is created that consists of the worker ID, the variable name of this data and the data itself in its native format. That is, each record has a field for a string, a float, a double etc. where all fields but one are empty. Admittedly, this does waste disc space but is still beneficial because one does not have to worry about data conversion in different programming languages or operating systems.

4. The migration table record is updated setting the state to "migration data stored".

As described above, all primitive data types (such as integer, float etc.) require just two field in a record, namely variable name and value. For more complex data types, such as arrays, collections, or specialized objects (e.g. composed result types) we have introduced a simple hierarchy: each record can set a *have children flag* and a *parent ID entry.* If the former is set this variable reflects some complex datastructure such as an array that contains children elements. These children elements then have the ID of the parent element stored in the parent ID field. This quite flexible design pattern allows for the mapping of almost any complex structure even between programming languages, as long as the worker's implementations are aware of the used variables.

Job Resumption

The second part of the migration procedure is the recovery of the original state and resumption of the algorithm on a different machine. To allow for this the target worker (that has required the suspended job) must be capable of (a) restoring the state of the source worker and (b) seamlessly continue execution.

The main prerequisite for a worker to be able to restore another worker's state is that it uses the same variable names internally, or at least knows how to map them onto the actual variables used. At initialization of a worker the *core data list* (see previous section) is created and the process of resuming is as follows:

1. The migration table record with the ID of the source worker is updated setting the state to "migration data transferring to target".

2. The *core data list* (see above) is traversed and for each object the "load" method is called.

3. To check whether the core data list is compatible with the data stored in the database, after each single transmission each item in the core data

list that has been loaded is flagged with a "load success" flag. Similarly, each data element in the database is also flagged with a "restored" flag.

4. After the core data list has been traversed the worker checks whether all elements in the list have set the "load success" flag and all elements at the database have set the "restored" flag. If this check fails, the migration is terminated and the migration table record is updated setting the state to "migration failed" including a reference to the worker that failed the data transmission.

5. If the check succeeds the migration table record is updated setting the state to "migration successful" and migration data at the database is deleted.

6. Prior to the actual computation the worker checks whether all required data - for example spectra that might have been references - is available locally and possibly requests and loads missing data.

7. Finally, the worker needs to continue at the algorithmic entry point where computation continues.

5.5 QAD Grid Platform Services

A service in the QAD Grid platform refers to either a particular *computational* service (such as peak picking or converting data formats) or a *system* service (such as workflow execution/control or restarting of unstable workers). A service consists of three components:

- The service specification (*registration*) at the QAD Grid server that describes the identification string (see section 5.3.3) and the parameters needed to define a task for this service.
- An implementation of a worker able to handle tasks of this service.
- Some method to create new tasks of this service and submit them to the QAD Grid's job queue (see section 5.3.3).

In the QAD Grid system users can create their own (non-system) services by registering this service and developing an individual worker that can handle this particular type of task using the QAD Grid design principles (see section 5.4.1). A new service can also be fully integrated into the QAD Grid which enables central administration (for example worker injections, see section 5.3.3).

Once registered, a service is available within the QAD Grid and tasks of this type can be submitted which can be requested and handled by workers.

5.5.1 Service Registration

The registration process of a new service announces the new type at the QAD Grid servers and defines its input and output, important for the usage in workflows (see section 5.6). It consists of the following steps:

1. Through a web-based form a registered user can request a new service by entering detail information about the service. This includes a general description and a unique identification string of this service. This

identification string will be used later to match this service with workers and tasks. Further, description and specification of the input parameters used and the output format have to be specified.

2. The service request is reviewed by a QAD Grid administrator and either accepted or denied.

3. In case of a positive decision that new service is automatically inserted into the QAD Grid database.

5.5.2 Service Integration

The QAD Grid system allows so-called *worker injections* (see section 5.3.3). This means, if triggered by some event (e.g. new tasks of a particular kind are available), the QAD Grid server can log into some client machine and start a worker on that machine able to handle this task.

To enable this feature a service needs to be integrated into the QAD Grid system. Integration of a service requires the following steps:

- The service must be successfully registered (see above) at the system.

- Each worker implementation needs to be registered at the QAD server - again, through a web-form. This includes

  - description,
  - version number,
  - operating system(s) this worker runs on (e.g. Linux or Windows),
  - hardware platform(s) this worker runs on (e.g. Intel or Cell/BE),
  - 3rd party library dependencies,
  - class path,
  - command-line to start this worker.

  Each 3rd party library used by this worker must be available at the QAD Grid server. This is to avoid redundant storage of libraries within the Grid and therefore multiple transfer of files to a client. Further, by providing verified libraries through the QAD Grid server no misuse can happen. Missing libraries need to be requested and will be integrated into the system by an QAD Grid administrator.

- The actual implementation of this worker needs to be available at the QAD Grid server zipped into one archive. This zip archive is also transmitted through the web-form (see above). When a worker is injected into a client this zip archive is transferred and unzipped at the target directory (see section 5.3.3).

5.5.3 Task Submission

Submission of tasks to the QAD Grid is done by calling a stored procedure at the database. This call needs to include the task id string and the task's parameters. The stored procedure then checks

- whether the task's id string is valid

- whether the sender is allowed to submit new tasks

If the checks are positive the task is inserted into the system's task queue.

5.5.4 QoS / Service Management

Service management in the QAD Grid means

1. controlling and ensuring availability of (integrated) core services
2. controlling and ensuring availability of Grid resources
3. allowing new services to be *hot deployed* into the Grid system

Additional Quality of Service (QoS) management ensures not only the availability of a service but also the quality of the available services and resources. Effective and efficient QoS management is critical for a service grid. Another crucial point in delivering high quality services is that enough reliable workers for a particular service type are available. To ensure this two things must be monitored: (a) the actual number of available workers and (b) the quality of the available workers. Recall that a service is directly dependent on the workers performing the tasks of a particular service type. Therefore we have developed a service management architecture that

- Periodically evaluates available workers
- Periodically checks the Grid's state
- Can dynamically adjust the number of workers available in the Grid, bases on the global Grid state
- Enables the creation of *virtual private Grids*
- Allows task priorization
- Allows integration of hot deployed services

The details of this architecture are described in the following paragraphs.

5.5.5 Worker Evaluation

A worker is evaluated based on its performance characteristics and its connection history. That is, the better the CPU speed (CPU, GHz), the more free memory available (MEM_{avail}, GByte) and the better the connection speed ($CONN_{up}, CONN_{down}$, KByte/s), the higher the score (S) of a worker W:

$$S_W = CPU + MEM_{avail} + \frac{CONN_{up} + CONN_{down}}{2}$$

5.5.6 Evaluating the Grids State

The global Grid state is determined by:

- Job waiting time ($JWT_{sertype}$, seconds): average time between submission and start of the last 20 jobs of service type *sertype*. We define a value above 60 seconds as insufficient
- Job queue length ($JQL_{sertype}$): number of submitted but not started tasks in the queue of service type *sertype*. We define five unstarted jobs for each available worker as barely acceptable.

- Database load (DBL, in percent)

We can therefore compute the Grid's state (GS) for each particular service type *sertype*:

$$GS_{sertype} = \frac{JWT_{sertype}}{60} \cdot \frac{JQL_{sertype}}{5 \cdot workers_{sertype}} \cdot DBL$$

where $workers_{sertype}$ is the number of currently available workers of type *sertype*. The total Grid state is simply the average state.

With this measurement in hand we can now evaluate the Grid's state and identify possible bottlenecks with respect to the single services available in the Grid. If a value exceeds "1" for a particular service this triggers the start of new workers of this service type and is described in the next section.

5.5.7 Dynamically Adjust Number of Workers

As described in section 5.3.3 the QAD system is able to start new workers at some target machine registered in the QAD Grid. When a performance value of a service exceeds "1" the QAD Grid server determines how many additional workers are needed. Based on the offline client monitoring (see section 5.3.4) the available machines are ranked and worker instances are injected into the top ranked machines.

On the other hand, if a performance value is close to "0" the QAD Grid server determines if workers - that are controlled by the QAD Grid - can be shut-down.

5.5.8 Virtual Private Grids

A virtual private grid (VPG) is a sub-set of available QAD Grid workers that are exclusively assigned to a particular user group. This restricted access can be set either by a QAD Grid administrator for nodes that are controlled by the QAD Grid or by the owners of the client nodes. These nodes will then only compute tasks that were submitted by a member of this particular user group.

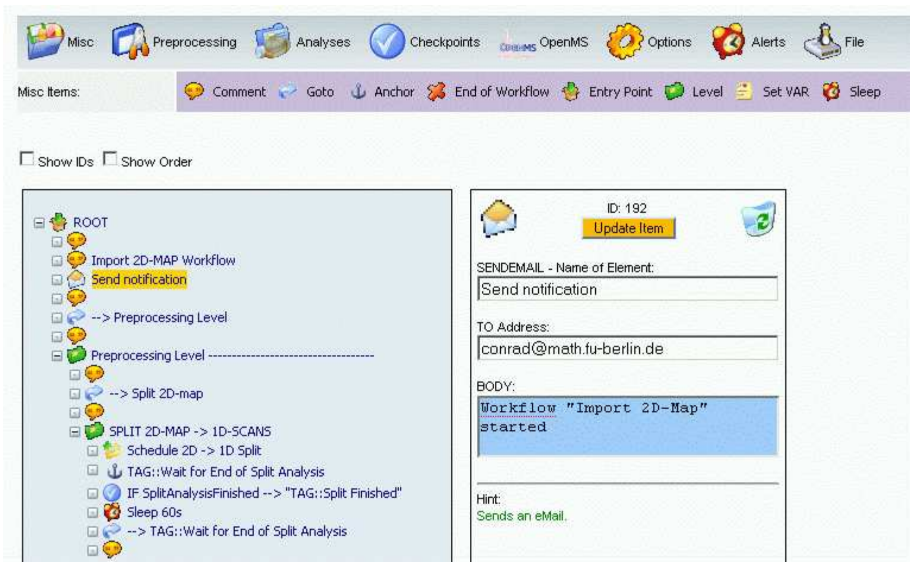
The advantage of a VPG is that it can be dedicated to a particular project where it is not slowed down by computing foreign tasks but still can use all features of the QAD Grid. These Grids are especially well suited for hot deployment of services.

5.5.9 Task Prioritization

Each task to be computed by the QAD Grid has a standard priorization. When a worker requests a task the task list is first ordered by priority and then by submission time. This means, a task with high priority will be computed before a standard priority task even if that task was submitted earlier. This tool allows the quick execution of *important* tasks even if the task queue contains many entries.

5.6 QAD Grid Workflows

As introduced in section 5.1.2 (at page 109) a workflow is the orchestration of tools and services. Built together from simple components it is well suited to



ROOT
Import 2D-MAP Workflow
Send notification
--> Preprocessing Level
Preprocessing Level --------------------------------
--> Split 2D-map
SPLIT 2D-MAP -> 1D-SCANS
Schedule 2D -> 1D Split
TAG::Wait for End of Split Analysis
IF SplitAnalysisFinished --> "TAG::Split Finished"
Sleep 60s
--> TAG::Wait for End of Split Analysis

SENDEMAIL - Name of Element: Send notification

TO Address: conrad@math.fu-berlin.de

BODY: Workflow "Import 2D-Map" started

Hint: Sends an eMail.

Figure 5.6.10: An example workflow shown on the left. On the right hand side the parameters of the selected icon (highlight in yellow) are shown. The gray and purple bars at the top are the toolbars containing the available workflow components.

create large pipelines to solve complex problems, e.g. a combination of many preprocessing steps followed by a multi-level analysis of data. A big advantage of workflows over manual execution of different tools is that a workflow is designed once and then applied to many different datasets (even in parallel).

For a workflow system we need two basic components: (a) the workflow designer and (b) a workflow execution module, which are described below. In the QAD Grid system each workflow component is either (1) a QAD Grid service which will be handled by a worker (e.g a particular analyses), (2) a basic programming language function (such as an if/then condition or a variable assignment) or (3) an extended workflow function (such as sending an email or deleting a file).

In the QAD grid system workflows are modeled as rooted directed graphs. Since we allow (conditional) jumps (see below) it can contain cycles. Figure 5.6.10 shows a small section of an example workflow. The left hand side of the figure shows the workflow in the workflow editor. Each icon represents one component of the workflow. At the right hand side, the parameters of the selected item (dyed in yellow) are shown: since this component is a "send email" item one needs to specify the receiver address and a mail body. Other items would presumably have more parameters.

An important concept in the QAD Grid workflow system is the usage of *user defined* variables. These variables allow the assignment of values to workflow variables at execution time. That means, a workflow can be designed and variables, such as the dataset to be analyzed or particular algorithm parameters, can be set by the user when the workflow is executed.

5.6.1 Workflow Elements

The following paragraphs describe the elements that can be used to build a workflow.

Services

Each registered service (see section 5.5.1) can be used in a workflow. During the service registration process the needed input parameters and the output format are defined. When a service is included in a workflow for each of these input parameters a source must be specified. This can be a fixed parameter (e.g. a numeric value for an algorithm), the result of a database query, such as the result of a preceding workflow step or a user defined parameter that will be fixed at execution time (see below).

In addition to these Grid services there exist further system services that require client/server interaction. For example, services that copy files from a server to a client.

Programming Language Elements

A QAD Grid workflow can contain so-called *programming language elements*. These are constructs known from imperative programming languages. The QAD Grid system knows the following constructs:

Variables: The QAD Grid workflow system provides two kinds of variables: (a) workflow variables that are initialized and modified during the workflow design phase and (b) user defined variables that are initialized by the user at the time of execution and can be modified during workflow execution. Usually at least one user defined variable is used within a workflow that defined the dataset that is used.

Assignment statements: Each variable can be assigned a value. Internally the value is represented as a string object and converted to numbers if required during workflow execution.

Evaluation of expressions: Simple expressions (such as basic arithmetics or relational operators) can be evaluated at runtime. For example a variable “VAR1” can be assigned a value “VAR2 * 5” that would be first evaluated to five times the current value of the variable “VAR2” and then assigned to “VAR1”.

Unconditional branching statements: This is commonly known as *jump* or *goto*. This item has a parameter that contains the target item the workflow execution process will jump to. Only special items (tags) can be used as jump targets.

Conditional branching statements: When the workflow execution process hits this item type it first evaluates the condition parameter which must evaluate to “false” or “true”. For example a valid condition might be “VAR1 $\geq$ 5”. For each outcome (“true or false”) a jump target can be defined where the process will branch to. If none is given the process will continue with the next item.

There are also higher conditional branching elements available for the user, such as readily implemented checks whether all jobs of a particular kind of this particular workflow are finished or whether a task finished with an error. This gets very handy for example if during pre-processing a 2D spectrum is to be split up into its 1D components. Then, this item can be used to check whether all of this splitting jobs have finished so the next workflow stage can be started.

Looping statements: Using the above elements a simple loop can be created:

1. Define a counting variable, say *VAR1*, and initialize it to “0”
2. Define a start tag (jump target), say *TAG::loop_start*
3. Use the conditional branching to evaluate “VAR1 $\leq$ x”, where “x” is the number of desired loop cycles - define the jump target for a negative evaluation to *TAG::loop_end* (see below)
4. Define the elements to be executed within the loop
5. Increase *VAR1*
6. Jump to *TAG::loop_start*
7. Define an end tag (jump target), say *TAG::loop_end*

Server-side Functions

Server-side functions are functions that are executed on the server. At the moment the following functions can be used within a workflow:

Sleep: Suspends the workflow execution for a given time, for example to wait for an analysis to finish.

Send email: Sends an email. This can be used for example to report that a workflow has finished.

Send SMS: Sends a SMS via a SMS gateway provider[20]. This feature can be used to report urgent system messages.

Directory and file I/O: During execution the workflow control process has full control over an exclusively assigned directory. In this directory it can create, delete, copy and move files and directories and read from/write to text files.

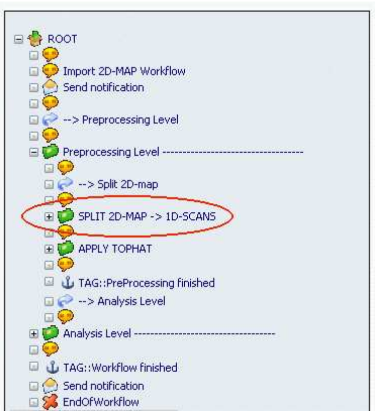
Log entries: During workflow execution entries in the global log file can be created for debugging purposes.

5.6.2 Workflow Designer

The workflow designer is a graphical online development interface accessible through the QAD Grid system. It allows easy creation and modification of workflows. Its core components (as shown in Figure 5.6.10) are the toolbar (top), the workflow viewer (left) that displays the workflow as a tree and the parameter viewer (right).

To add a new item to a workflow it is simply dragged from the toolbar and dropped at the workflow tree at the position it is supposed to appear (see Figure 5.6.11). To correct its position an item can also be dragged and dropped within a workflow tree. To modify the parameters of an item it has to be selected by clicking on it. Then, the available parameters are shown in the parameter view on the right hand side (see figure 5.6.10 for the parameters of the “Send email” item). Here, the parameters can be changed or the item can be removed from the workflow by clicking on the trash icon on the upper right corner of the parameter view.

[20]Currently we are using the Clickatell™service (`http://www.clickatell.com`) because it allows for the control of the User Data Header and supports the SMSJ-SMS library for Java (see `http://smsj.sourceforge.net`) and many other programming languages.



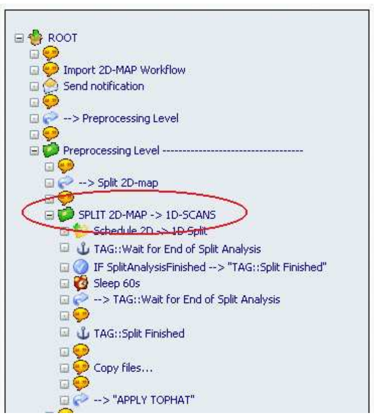


Figure 5.6.12: The same example workflow collapsed (left) and expanded (right).

For better visual partitioning of complex workflows levels can be introduced. Each level can contain an unlimited number further sub-levels and items and can be collapsed and expanded (see Figure 5.6.12).

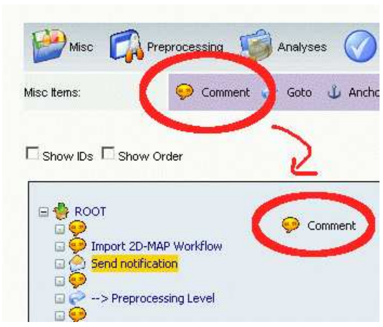


Figure 5.6.11: Screenshot from the proteomics.net workflow designer: An item is dragged from the toolbar and dropped at a workflow.

Workflow Check

Most of the item input parameters cannot be input freely but selected from a given valid set. For example, jump targets can only chosen from a list of targets that really exist at the moment of creation. Consequently, a jump target can only be deleted when it is not referenced by an item within the workflow. Consequently, only a syntax check for the free text parameters has to be performed.

5.6.3 Workflow Execution

Workflows are executed by specialized services running at the QAD Grid server. When running a workflow the user first needs to specify the *user defined variables* required in this workflow. The execution request and the variable values are then inserted into the database. Similar to the job execution (see section 5.3.3 a workflow execution service then marks this workflow as “in progress” and requests / loads the (graph) structure. Starting with the root node the following steps are performed following the *state machine paradigm* (see e.g. (Gill, 1962)):

1. Depending on the current node the matching node type handler is called with the input parameters.

2. The node handler

    - checks the input parameters
    - evaluates variables and conditions if necessary
    - performs required actions - usually this means that tasks are created and scheduled.
    - returns the next node to execute: either the succeeding tree node or a jump target

3. The workflow execution service stores the state of all current variables and the new active node to the database. This allows for resuming from this point after a system crash.

4. The workflow execution service sets the returned node to active and starts over with step (1).

Experiments have shown that the workflow execution service is idle most of the time during execution. Therefore, we have implemented this service as a multi-threaded service: The main thread handles all database requests including scanning the database for new workflows to execute. If a new request is found and the overall system load is not too high a child-thread is created that handles this workflow. The main advantage of this approach is that we save connections to the database, since only the main thread needs to have an active database connection. This is beneficial since each open connection slows down the database server (see e.g. (Huddleston et al., 2006)).

5.7 Related Work

As mentioned in the previous sections, most systems that are used for building distributed computing systems or computational grids have evolved to quite complex software frameworks. In Table 5.7.1 we list some exemplary Grid projects that are currently (2007/2008) active and have been studied by us[21]. There are roughly three categories these projects fall into:

1. **e-Infrastructure**: This refers to a research environment in which a researcher has shared access to scientific facilities (such as data, computing or sensors), regardless of their type and location in the world.

2. **Middleware**: A middleware connects software applications enabling exchange of data. Thus, it organizes and integrates the resources in a Grid. One of its main purpose is to automate the required *machine to machine* negotiations, such as negotiating the exchange of resources on behalf of Grid users and resource providers. It also provides the core foundation (basic services) for grid applications including areas such as: security, resource management, information services and data management.

3. **Application:** These are projects within the context of specific scientific fields that are devoted to explore and harness grid technology. Depending on its state (*fully operational* vs. *experimental*) these projects are called application or *testbed*.

Since there exists neither *the* exemplary Grid system nor a standard for Grid systems (although systems like Globus seem to become the de-facto standard) we will not describe any system in great detail nor give a detailed comparison to our system[22]. Rather, we use the table to illustrates two things: (a) there is a need for distributed computing systems in many application areas and (b) there is a large variety of systems with different scales, aims, approaches and technological foundations. However, as far as our analyses have shown none of them manages to provide a powerful and at the same time easy

[21]Another even more exhaustive list can be found here (GridInfoware, 2008)).

[22]For a detailed comparison of four large Grid systems see e.g. (Asadzadeh et al., 2006) and references therein.

to set-up system that can be used in an (quasi) ad-hoc manner, as our system is designed to operate. Further, there does not seem to be a fully equipped framework that provides all layers such as middleware, workflow system, Grid monitor and web portal. All of these layers have been incorporated in the system introduced in this thesis.

Table 5.7.1: Overview of several Grid projects

Project	**Category**	**Description**
@neurIST	Application	Grid platform for management, integration and processing of data associated with cerebral aneurysm and subarachnoid haemorrage.
ACGT	Application	Grid platform to support exchanges of clinic and genetic information with a particular focus on breast cancer data.
BioinfoGRID	Middleware	This project aims to connect European computer centers to offer computational services especially in the area of Bioinformatics.
BIOPATTERN	Application	Within this project the members want to create a pan-European platform enabling sharing and analysis people's bioprofiles. This should allow to combat major diseases such as cancer and brain diseases.
Condor	Middleware	Condor is a specialized workload management system for compute-intensive jobs. Similar to other batch systems, Condor provides a job queueing, several scheduling policies and priority schemas, resource monitoring and management. The Condor-G extension is fully interoperable with resources managed by the Globus system.
DataGrid	Application	This was one of the first projects (and predecessor to the EGEE project) that aimed to enable intensive computation and analysis of shared large-scale databases across widely distributed scientific communities.
DataMining Grid	Application	This consortium develops generic and sector-independent data-mining tools and services for the grid.
D-Grid	Infrastructure	The German national Grid initiative wants to establish a sustainable Grid infrastructure for e-Science in Germany by (a) strengthening existing Grid user communities and (b) integrating and extending existing but dispersed middleware testbeds.
DutchGrid	Application	The DutchGrid is the platform for Grid Computing and Technology in the Netherlands.

Continued on Next Page...

Table 5.7.1 – Continued

Project	Category	Description
Edutain@Grid:	Application	Edutain@Grid seeks to transfer techniques from Grid computing to networks used in massive online games through development of a Grid-based framework allowing responsive and interactive applications.
EGEE	Infrastructure	The Enabling Grids for E-sciencE (EGEE) project is funded by the European Commission and aims to build on recent advances in grid technology and develop a service grid infrastructure which is available to scientists 24 hours-a-day.
EGRID	Infrastructure	An Italian GRID infrastructure for finance and economic research.
EUChina	Infrastructure	This Grid aims to facilitate scientific data transfer and processing between Europe and China.
g-Eclipse	Application	The g-Eclipse project aims to build an integrated workbench framework to allow access to existing Grid infrastructures. It is built on top of the Eclipse framework and provides tools to customize Grid user's applications, to manage Grid resources and to support the development cycle of new Grid applications.
gLite	Middleware	Lightweight middleware for Grid Computing. gLite integrates components from current middleware projects, such as Condor and the Globus Toolkit, as well as components developed for the LCG project.
Globus	Middleware	The Globus Project is developing fundamental technologies needed to build computational grids, which enable software applications to integrate a variety of resources. The project includes investigations of security, resource management, communication protocols, data management mechanisms, etc.
GRIDCC	Application	GRIDCC will extend the commonly used batch access to distributed computational and storage resources by including access to and control of distributed instrumentation.
Health-e-Child:	Application	is an integrated platform for European paediatrics based on a grid-enabled network of leading clinical centers.
Interactive European Grid	Infrastructure	The goal of the project is to deploying and operate an inter operable production-level e-Infrastructure for demanding interactive applications. Distributed parallel computing is based on MPI.
myGrid	Application	The myGrid projects puts emphasis on the information Grid and provides middleware layers tailored to the need of scientists in the area of bioinformatics.

Continued on Next Page...

Table 5.7.1 – Continued

Project	**Category**	**Description**
NorduGrid	Middleware	Also known as *Advanced Resource Connector* (ARC) they provide a reliable implementation of the fundamental Grid services, such as information services, resource discovery and monitoring, job submission and management, brokering and data management and resource management.
SEE-GRID-2	Infra-structure	The successor of the South-East Europe (SEE) GRID aims to further advance the existing infrastructure and services, to further strengthen scientific collaboration and cooperation among participating SEE communities, and ultimately achieve sustainability for regional and national e-Infrastructures.
TeraGrid	Appli-cation	The aim of this project is to create an integrated, persistent computational resource. Currently, TeraGrid resources include more than 750 teraflops of computing capability and more than 30 petabytes of online and archival data storage. Researchers can also access more than 100 discipline-specific databases. They claim to be the world's largest, most comprehensive distributed cyberinfrastructure for open scientific research.
UniGrids	Appli-cation	This project develops a grid service infrastructure based on the Open Grid Services Architecture (OGSA) and on the UNICORE grid software. The goal of UniGrids is to develop translation mechanisms, such as resource ontologies, to interoperate with other OGSA compliant systems. At the same time UniGrids will target grid economics by developing a SLA framework and cross-grid brokering services. The scientific focus is set to be in the areas of biomolecular and computational biology, energy, geophysical depth imaging by oil companies and reactor safety.

Chapter 6

proteomics.net - Product-oriented Case Studies

Contents

This chapter will describe our *proteomics.net* platform. This platform was developed with three goals in mind:

- Create a reference implementation of the algorithms for the analysis of mass spectra described in chapter 3.
- Create a reference implementation of the Grid approach specified in chapter 5 to enable handling and analysis of mass data.
- Create an intuitive web-based user interface to allow easy usage of this platform for us and our collaborators enabling research as described in chapter 4.

During this work many other positive side effects could be detected, such as:

Organization of data and results: Often a project lacks sufficient organization, documentation and storage of source and resulting data or parameters used in different analyses. The presented platform does solve these problems in the domain of (mass spectrometry based) proteomics research. Once the raw data is integrated into the system each analysis performed is fully documented and each result is stored and can be traced back to the data and parameter-sets used.

Comparison of results: During the course of a (data centered) project usually many different algorithms are developed or tested. Additionally, most algorithms can be parameterized in many ways often with large variations in the results. The platform presented here cannot only organize the results depending on algorithms and parameters and thus enable reasonable comparisons but also provides consistent visualization.

Exploring parameter space: Many algorithms depend on a lot of input parameters. Determining and optimizing these parameters for a particular dataset or certain purpose is not always possible. The presented platform in combination with the QAD Grid workflow system does not only allow to create different parameter-sets but also starts the desired algorithms with these input parameters on many different machines in parallel and monitors the execution and can also evaluate the results and send reports to the user.

Rapid Grid application development: Thanks to the worker concept and available web-services embedding of almost any available program code becomes possible quite quickly - even on specialized hardware.

6.1 Available Services

The *proteomics.net* platform offers the following services:

- Web portal for (MS-based) proteome research
  - Fostering organization, exchange and provision of (raw) data and (intermediate) results)
  - Provision and easy usage of tools and algorithms (section 6.2.1)
  - Providing of web services to use available algorithms in local applications (section 6.2.2)
- Distributed computing (Grid) services
  - Enabling processing and analysis of proteomics mass data faster than real time
  - Allowing integrating of specialized hardware, such as IBM's Cell/BE processor in Sony's Playstation 3 for compute intensive tasks (section 6.2.4)
  - Enabling non-Grid applications to use the power of a distributed computing environment (section 6.2.5)
  - Easy creation and integration of Grid applications (section 6.2.6)

6.2 Case Studies

6.2.1 Software as a Service: Using the Web Based Platform Services

This section describes the main concept of the *proteomics.net* platform: host, operate and provide algorithms that can be used over the internet with user-specific data. We will give a brief example in the proteomics context that shows the benefits of this concept for an user that does not have the needed IT infrastructure to solve a particular problem. It is generally suited for users that have little interest or capability in software deployment, but do have substantial computing needs.

Background

Performing an arbitrary data analysis requires of course (1) the data to be analyzed and (2) a suitable analysis algorithm (computer program). Often there arise problems concerning (2): installing, maintaining and running a software package is not always a trivial task. Problems like insufficient hardware (e.g. memory) or an incompatible software platform (e.g. operating system or libraries) can render the use of a particular software impossible.

Our Approach: Software as a Service

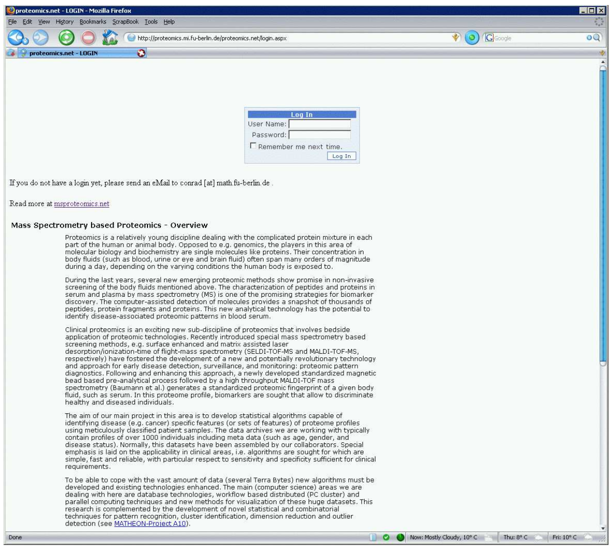
With the *proteomics.net* platform we have chosen to provide our algorithms using the *Software as a Service* (SaaS) approach. This means, that the algorithms are hosted on our IT infrastructure and can be used through a web based user interface. To give an example we briefly sketch the steps a user needs to execute to perform a peak picking analysis (as described in section 3.4:

1. Open the platform's web portal in a web browser
2. Log into the platform (Figure 6.1(a))
3. Upload the data to be analyzed (e.g. by FTP)
4. Start a new peak picking analysis (figure 6.1(b))
5. Select the dataset and parameter-set to be used (figure 6.2(a))
6. Assign a name and description (Figure 6.2(b))
7. Start the analysis

After the peak picking analysis is finished the results will be available for inspection, further usage or export/download. Note that the results are automatically

- archived in the system
- linked to the raw data the peak picking was performed on
- linked to the parameter-set used

Note that the data analysis is automatically scheduled to the best matching computing ressources available.


proteomics.net - LOGIN - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/login.aspx
proteomics.net - LOGIN
Log In
User Name:
Password:
Remember me next time.
Log In

If you do not have a login yet, please send an eMail to conrad [at] math.fu-berlin.de .

Read more at msproteomics.net

Mass Spectrometry based Proteomics - Overview

Proteomics is a relatively young discipline dealing with the complicated protein mixture in each part of the human or animal body. Opposed to e.g. genomics, the players in this area of molecular biology and biochemistry are single molecules like proteins. Their concentration in body fluids (such as blood, urine or eye and brain fluid) often span many orders of magnitude during a day, depending on the varying conditions the human body is exposed to.

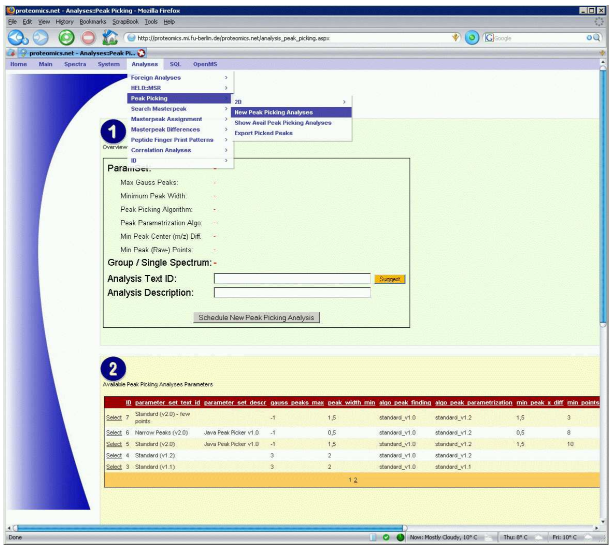
During the last years, several new emerging proteomic methods show promise in non-invasive screening of the body fluids mentioned above. The characterization of peptides and proteins in serum and plasma by mass spectrometry (MS) is one of the promising strategies for biomarker discovery. The computer-assisted detection of molecules provides a snapshot of thousands of peptides, protein fragments and proteins. This new analytical technology has the potential to identify disease-associated proteomic patterns in blood serum.

Clinical proteomics is an exciting new sub-discipline of proteomics that involves bedside application of proteomic technologies. Recently introduced special mass spectrometry based screening methods, e.g. surface enhanced and matrix assisted laser desorption/ionization-time of flight-mass spectrometry (SELDI-TOF-MS and MALDI-TOF-MS, respectively) have fostered the development of a new and potentially revolutionary technology and approach for early disease detection, surveillance, and monitoring: proteomic pattern diagnostics. Following and enhancing this approach, a newly developed standardized magnetic bead based pre-analytical process followed by a high throughput MALDI-TOF mass spectrometry (Baumann et al.) generates a standardized proteomic fingerprint of a given body fluid, such as serum. In this proteome profile, biomarkers are sought that allow to discriminate healthy and diseased individuals.

The aim of our main project in this area is to develop statistical algorithms capable of identifying disease (e.g. cancer) specific features (or sets of features) of proteome profiles using meticulously classified patient samples. The data archives we are working with typically contain profiles of over 1000 individuals including meta data (such as age, gender, and disease status). Normally, this datasets have been assembled by our collaborators. Special emphasis is laid on the applicability in clinical areas, i.e. algorithms are sought for which are simple, fast and reliable, with particular respect to sensitivity and specificity sufficient for clinical requirements.

To be able to cope with the vast amount of data (several Terra Bytes) new algorithms must be developed and existing technologies enhanced. The main (computer science) areas we are dealing with here are database technologies, workflow based distributed (PC cluster) and parallel computing techniques and new methods for visualization of these huge datasets. This research is complemented by the development of novel statistical and combinatorial techniques for pattern recognition, cluster identification, dimension reduction and outlier detection (see MATHEON-Project A10).

Done
Now: Mostly Cloudy, 10° C
Thu: 8° C
Fri: 10° C


proteomics.net - Analyses::Peak Picking - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/analysis_peak_picking.aspx
proteomics.net - Analyses::Peak Pi...
Home
Main
Spectra
System
Analyses
SQL
OpenMS
Foreign Analyses
HELD::MSR
Peak Picking
Search Masterpeak
Masterpeak Assignment
Masterpeak Differences
Peptide Finger Print Patterns
Correlation Analyses
ID
2D
New Peak Picking Analyses
Show Avail Peak Picking Analyses
Export Picked Peaks
1
Overview
Suggest
Schedule New Peak Picking Analysis
2

ParamSet:	-
Max Gauss Peaks:	-
Minimum Peak Width:	-
Peak Picking Algorithm:	-
Peak Parametrization Algo:	-
Min Peak Center (m/z) Diff.	-
Min Peak (Raw-) Points:	-
Group / Single Spectrum:	-
Analysis Text ID:	
Analysis Description:	

Available Peak Picking Analyses Parameters

	ID	parameter_set_text_id	parameter_set_descr	gauss_peaks_max	peak_width_min	algo_peak_finding	algo_peak_parametrization	min_peak_x_diff	min_points
Select	7	Standard (v2.0) - few points		-1	1,5	standard_v1.0	standard_v1.2	1,5	3
Select	6	Narrow Peaks (v2.0)	Java Peak Picker v1.0	-1	0,5	standard_v1.0	standard_v1.2	0,5	8
Select	5	Standard (v2.0)	Java Peak Picker v1.0	-1	1,5	standard_v1.0	standard_v1.2	1,5	10
Select	4	Standard (v1.2)		3	2	standard_v1.0	standard_v1.2		
Select	3	Standard (v1.1)		3	2	standard_v1.0	standard_v1.1		

1 2

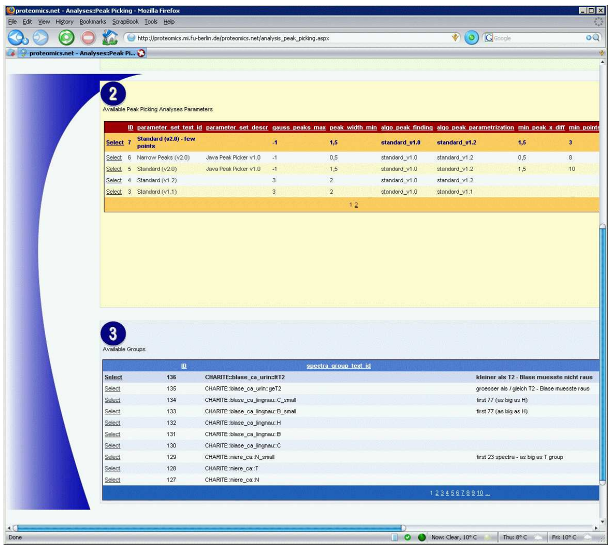
Figure 6.2.1: Screenshot of the proteomics.net framework. Left: Login screen; right: Starting a new peak picking analysis

6.2.2 Web Services: Using the Platform Services in Local Programs

A second possibility to use the algorithms provided by the *proteomics.net* framework is by using *Web Services*. Web services can be seen as APIs (application programming interface) accessible over the Internet. By using them, algorithms can be executed on a remote system hosting the requested services. This means that users can write their own software that includes calls to an algorithm hosted by the *proteomics.net* framework.

All algorithms and services available at the *proteomics.net* platform are accessible via system independent web-services from outside the platform. This means (authorized) local stand-alone software (written in almost any programming language) can

- query the platform (e.g. get status information),
- read and send data or results,
- submit new jobs, that is, use the algorithms available at the platform.


proteomics.net - Analyses::Peak Picking - Mozilla Firefox
http://proteomics.mi.fu-berlin.de/proteomics.net/analysis_peak_picking.aspx

2
Available Peak Picking Analyses Parameters

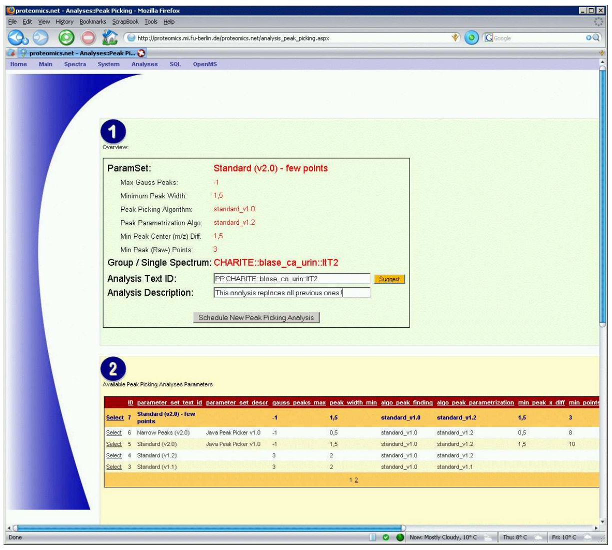
	ID	parameter_set_text_id	parameter_set_descr	gauss_peaks_max	peak_width_min	algo_peak_finding	algo_peak_parametrization	min_peak_x_diff	min_points
Select	**7**	**Standard (v2.0) - few points**		**-1**	**1,5**	**standard_v1.0**	**standard_v1.2**	**1,5**	**3**
Select	6	Narrow Peaks (v2.0)	Java Peak Picker v1.0	-1	0,5	standard_v1.0	standard_v1.2	0,5	8
Select	5	Standard (v2.0)	Java Peak Picker v1.0	-1	1,5	standard_v1.0	standard_v1.2	1,5	10
Select	4	Standard (v1.2)		3	2	standard_v1.0	standard_v1.2		
Select	3	Standard (v1.1)		3	2	standard_v1.0	standard_v1.1		

1 2

3
Available Groups

	ID	spectra_group_text_id	
Select	**136**	**CHARITE::blase_ca_urin::ltT2**	**kleiner als T2 - Blase muesste nicht raus**
Select	135	CHARITE::blase_ca_urin::geT2	groesser als / gleich T2 - Blase muesste raus
Select	134	CHARITE::blase_ca_lingnau::C_small	first 77 (as big as H)
Select	133	CHARITE::blase_ca_lingnau::B_small	first 77 (as big as H)
Select	132	CHARITE::blase_ca_lingnau::H	
Select	131	CHARITE::blase_ca_lingnau::B	
Select	130	CHARITE::blase_ca_lingnau::C	
Select	129	CHARITE::niere_ca::N_small	first 23 spectra - as big as T group
Select	128	CHARITE::niere_ca::T	
Select	127	CHARITE::niere_ca::N	

1 2 3 4 5 6 7 8 9 10 ...


proteomics.net - Analyses::Peak Picking - Mozilla Firefox
http://proteomics.mi.fu-berlin.de/proteomics.net/analysis_peak_picking.aspx
Home
Main
Spectra
System
Analyses
SQL
OpenMS

1
Overview:

ParamSet:	Standard (v2.0) - few points
Max Gauss Peaks:	-1
Minimum Peak Width:	1,5
Peak Picking Algorithm:	standard_v1.0
Peak Parametrization Algo:	standard_v1.2
Min Peak Center (m/z) Diff:	1,5
Min Peak (Raw-) Points:	3
Group / Single Spectrum:	CHARITE::blase_ca_urin::ltT2
Analysis Text ID:	PP CHARITE::blase_ca_urin::ltT2
Analysis Description:	This analysis replaces all previous ones !

Suggest
Schedule New Peak Picking Analysis

2
Available Peak Picking Analyses Parameters

	ID	parameter_set_text_id	parameter_set_descr	gauss_peaks_max	peak_width_min	algo_peak_finding	algo_peak_parametrization	min_peak_x_diff	min_points
Select	**7**	**Standard (v2.0) - few points**		**-1**	**1,5**	**standard_v1.0**	**standard_v1.2**	**1,5**	**3**
Select	6	Narrow Peaks (v2.0)	Java Peak Picker v1.0	-1	0,5	standard_v1.0	standard_v1.2	0,5	8
Select	5	Standard (v2.0)	Java Peak Picker v1.0	-1	1,5	standard_v1.0	standard_v1.2	1,5	10
Select	4	Standard (v1.2)		3	2	standard_v1.0	standard_v1.2		
Select	3	Standard (v1.1)		3	2	standard_v1.0	standard_v1.1		

1 2

Figure 6.2.2: Screenshot showing the creation of a new peak picking analysis. Left: Select dataset and parameter-set to use; right: Assign an analysis name and description. Note: in the proteomics.net framework the two screens are on top of each other.

Concepts

The platform provides information about available web-services in the web-service definition language (WSDL). Using tools like Apache AXIS' "WSDL2Java"[1] from this WSDL code necessary Java classes and bindings can automatically be created to enable usage of web-services in a Java environment. To allow a local software usage of the web-services it must be authenticated and authorized (see section 5.3.1).

For the following example, let us assume that there exists a web-service called "get_current_worker()" that simply returns a string containing type and name of the workers being online and their client machines. Let us further assume that necessary Java classes have been created automatically by the *WSDL2Java* tool and for this service the client does not need to be authenticated. Then, the code needed for a (very simple) Java client to connect to

[1] see `http://ws.apache.org/axis/java/user-guide.html\#WSDL2JavaBuildingStubsSkeletonsAndDataTypesFromWSDL`

the platform, get the worker status information and display it can be written in less than 20 lines as the following listing 6.1 shows:

Listing 6.1: Minimal code for using the "get_current_worker()" web-service in Java.

```
1  import org.apache.axis.client.Call;
2  import org.apache.axis.client.Service;
3  import javax.xml.namespace.QName;
4  import de.fu_berlin.mi.proteomics.proteomics_net_webservices.*;
5
6  public class client_java {
7      public static void main(String[] args) {
8          try
9          {
10             Services_infoLocator loc = new Services_infoLocator();
11             Services_infoSoap port = loc.getservices_infoSoap();
12             System.out.println(port.get_current_worker());
13         }
14         catch(Exception e)
15             {System.out.println(e.getMessage());}
16     }
17 }
```

The actual (synchronous) call to the web-service happens in line 12. This line could also contain far more complex calls, for example including objects as input and output parameters. Thanks to the transformation services (e.g. within the WSDL2Java tool) parameters send to and received from the web-service are mapped to Java data types.

6.2.3 Integrating External Web Services on the Example of Protein Identification

Background

One key issue in proteomics is to identify proteins and characterize their expressions in cells. In mass-spectrometry based proteomics this is done by either peptide mass fingerprinting (PMF) of MS^1 spectra or by further fragmenting single peptides producing MS^2 spectra where (ideally) the amino acid sequence can be derived from.

The PMF approach (also known as protein fingerprinting) is an analytical technique for protein identification developed in the early 1990s. The basic idea is to digest an unknown protein of interest by a sequence specific protease (such as Trypsin). The set of resulting peptides (fragments) build a unique identifier (fingerprint) of the unknown protein based on this protease and subsequently compared to databases containing known fragmentation patterns for this protease. Obviously, the mass accuracy to which the peptides are measured plays a crucial role (Green et al., 1999).

In MS^2 spectra analysis peptides of interest - identified during a MS^1 run - are fragmented further in a collision cell to produce tandem (MS/MS, MS^2) mass spectra. Since fragmentation (usually) happens at the backbone peptide bonds, by putting together matching pieces (that result in the full peptide) and analyzing the point of rupture (in principle) determination of the amino acids gets possible. This approach is called *De Novo* sequencing (see e.g. (Ma et al., 2003; Halligan et al., 2005) and references therein). The second large class of algorithms for the identification problem is based on comparing the experimental spectrum against a database of theoretical spectra determined by in silico digestion and fragmentation of known proteins.

MASCOT and SEQUEST (Perkins et al., 1999; J et al., 1994) are examples of this approach which employ sophisticated statistical models to determine the similarity of experimental and theoretical spectra.

Both approaches suffer from the fact that peptide fragmentation is a complex (biochemical) process, so spectra generated from mass spectrometers are often significantly different from their theoretical counterparts. However, database approaches have been reported to work quite effectively when used with standard data. Unfortunately, performance deteriorates significantly in certain settings, such as high abundance of homologue proteins, lack of sequence data or present peptide modifications. In such scenarios *de novo* methods provide an invaluable alternative because they infer protein sequences without using existing sequence data, while additionally accounting for possible peptide modifications.

Available Projects

As mentioned in the previous section building a protein/peptide identification algorithm is a complicated task and many sophisticated scientific and commercial projects are available for this. These are available as stand-alone programs, web-sites or web-services. The integration of stand-alone programs is described in section 6.2.5 so in this example we focus on the integration of web-based services.

Approach

To integrate web-based services into the *proteomics.net* framework we extended the base QAD Grid worker (see section 5.4.1) by methods for using web-forms (GET & POST), parsing HTML code and using web-services. With this new methods it becomes possible to use web pages (such as the Mascot services) and web-services (such as Emboss' *emowse* service from the Helmholtz Open BioInformatics Technology initiative, based on (Pappin et al., 1993)).

To use these services at the *proteomics.net* platform we implemented a protein ID worker that

- takes the ID of a dataset (peaklist) available at the platform, the desired ID service and needed parameters as input,
- send the request to the chosen service,
- waits for the answer,
- parses the result and converts it to the format used in the *proteomics.net* framework,
- inserts the result into the database and links to the source data.

This ID worker not only enables the integration of the protein identification service into the *proteomics.net* platform, but also allows execution of many queries in parallel and full integration into the workflow system.

WHOA! ALL OF THIS NEW COMPUTER SURGICAL EQUIPMENT IS COMPLICATED! WHERE DID YOU TRAIN...HARVARD? COLUMBIA?
PLAYSTATION

6.2.4 Embedding of Specialized Hardware on the Example of a Playstation 3

Background

In the last years specialized multi-core hardware such as IBM's Cell/BE processor or Graphic Processing Units (GPUs) from AMD and NVIDIA were released to the consumer market. Although having a radically different hardware architecture opposed to the common desktop PC, they are really interesting for computational tasks because they are equipped with many (co-) processing units for calculations. A smaller version (seven opposed to 16 cores) of IBM's Cell/BE processor is built into Sony's Playstation 3 (PS3) and can be purchased for about 400 EUR (in December 2007).

At 3.2 GHz, each SPE gives a theoretical 25.6 GFLOPS of single precision performance which add up to about 190 GFLOPS. Since in a PS3 only six SPEs are available for calculations (one is deactivated and another one reserved for the OS) about 150 GFLOPs can be achieved. This is about 15 times faster than a common Intel desktop PC (single core, running at 3.6GHz).

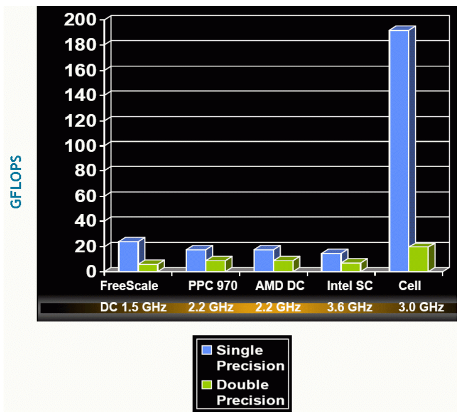


Figure 6.2.3: Cell performance values for eight SPEs. Note that in a PS3 only six SPEs are available for calculations. These benchmarks nicely reflect the optimization of the Cell towards single precision operations.

In a typical usage scenario, Linux is installed on the Playstation using a kernel extension by IBM. Using IBM's Cell SDK programs can be written in the C++ programming language that support the Cell's instruction set. This system will load the SPEs with these programs (similar to threads) which can then perform the given task in parallel.

Since the playstation has only a limited memory (RAM) of 512MB and reaches the good performance only in single-precision mode not all algorithms will run well on this technology. Further, each SPE has only 256KB of local memory that can be used for computations.

Porting Peak Picking

Motivated by these performance figures we ported the core of our reference worker from Java to C. This "Playstation BaseWorker" can communicate with the *proteomics.net* platform, request jobs and send back results.

Given the above mentioned limitations, we choose to implement and optimize the peak picking algorithm (see section 3.4.3) because the data can be processed linearly and easily splitted up into small packages.

The peak picking algorithm changed slightly opposed to the original version on a single core CPU and is now as follows:

1. Load a 1D spectrum into main memory (about 670KB in 54.000 x/y data points)
2. Split spectrum into pieces of size $min(256KB, \frac{spectrum\_size[KB]}{\#SPUs})$
3. Until pieces are available: Send pieces to SPEs to find peaks
4. Wait for all SPEs to finish
5. Merge results

Results

The results are shown in Table 6.2.4.

Machine	Test	# SPEs	Time [ms] per spectrum
Intel$^{\#}$	w/o fitting	1	8200
Intel$^{\#}$	w/ fitting	1	13300
PS3	w/o fitting	6	115
PS3	w/ fitting	1	2250
PS3	w/ fitting	6	750
QS20	w/ fitting	1	2100
QS20	w/ fitting	6	730
QS20	w/ fitting	12	410
QS20	w/ fitting	16	350

Table 6.2.1: This shows the PS3 performance tests. $^{\#}$: Standard desktop single-core PC at 3.2GHz

6.2.5 Integrating of Non-Grid Applications on the Example of Matlab Scripts and Linux Binaries

Background

As explained in the previous chapter, writing QAD Grid applications is not a very complex task but still, some modifications must be made to standard code. Thanks to the QAD Grid worker concept, there are ways to circumvent even this requirement as we will show below for algorithms existing as Matlab script and for binaries where the source code obviously cannot be modified. Besides the obvious advantages distributed computation offers, e.g. Load balancing, another very interesting point is service availability. That means, if we have many Matlab scripts providing the same service running on n different nodes in the network (and assume a stable QAD Grid server) then failure of up to $n-1$ nodes still does not harm the availability of this service, since the QAD Grid server will distribute incoming jobs to the remaining running nodes.

Approach

Matlab Scripts The scripting language Matlab is quite popular for quick coding of mathematical algorithms. A typical algorithm gets some input parameters and data and then computes and returns the results. To allow almost any user-defined matlab script to be run within a QAD Grid two things must be done: (a) a special Matlab worker must be started at the Grid nodes that will then run the user-defined Matlab scripts and (b) the function call of the user-defined Matlab script must be changed to take two more parameters. If the user-defined script is called from the Matlab worker these two additional parameters contain (1) a string "MCP" and (2) an object reference to the underlying QAD Grid worker. Through this the base QAD Grid worker functions (see section 5.4.1) can be used, such as writing log entries or change the worker's status. If the Matlab worker is started at a QAD Grid node it performs the following steps:

1. Start Matlab at the Grid node
2. Start the Matlab worker script
3. Within the Matlab worker:
   - Wait until a new Matlab task is available
   - Request the name of the user-defined Matlab script to be started
   - Request the parameters for the user-defined Matlab script
   - Start the user-defined Matlab script

The Matlab worker core that gets the parameters from the QAD Grid and starts the user-defined script is implemented as shown in listing 6.2:

Listing 6.2: Minimal code for using the a Matlab worker.

```
 1  params = jPROTEOMICSWORKER_MATLAB_MCP.get_next_task();
 2
 3  if( params ~= [] )
 4     % param1: name of script
 5     % param2 - param8: params for called script
 6     m_script = char( params(1) );
 7
 8     ...
 9
10     c = [m_script ' ( "MCP", jPROTEOMICSWORKER_MATLAB_MCP'];
11     for i = 2:9
12        if( ~ isempty( char(params(i)) ) )
13           c = [ c ', "' char( params(i) ) '" ' ];
14        end
15     end
16     c = [ c ' )'];
17
18     jPROTEOMICSWORKER_MATLAB_MCP.change_state( ['Processing Task ID: '
19                                                id ' - calling: ' c] );
20     c = strrep( c, '"', '''' );
21
22     % execute matlab script
23     eval( c );
24  end;
```

Line 1 requests a new task from the QAD Grid. If a new job is available the 9-dimensional array "params" contains the parameters of this task. The first entry in “params” contains the filename of the user-defined script, the remaining array entries contain the parameters. Line 10-16 show how the calling string “c” is built and that the first two parameters mentioned above are the “MCP” string and the object reference (line 10). Line 18 shows how the QAD Grid worker object can be used to communicate directly with the QAD Grid: here, the state string of the worker is changed, which will automatically be updated in the QAD Grid database. The Matlab built-in function “eval” is then used with the start string “c” as parameter to start the actual user-defined Matlab script.

Binaries The QAD Grid worker concept offers another far more general approach to enable non-Grid applications in the case where no source code is available. Using a *wrapper* worker almost any binary application can use the power of distributed computing. To achieve this, a standard base worker is available that offers the following:

- Authenticate at and integrate into the *proteomics.net* platform
- Request new tasks (and respective parameters) for the configured purpose
- Perform any system call or run any binary application
- Process results and transmit them to the *proteomics.net* platform

To give a concrete example listing 6.3 shows how we use a wrapper worker to start a binary (“tophat”, line 47-51), and process the results (copy to some destination, line 57).

Listing 6.3: Core of wrapper worker to start tophat filter binary.

```
1   public static void main(String[] args) throws Exception {
2
3         ...
4
5         % initialize object
6
7         PROTEOMICSWORKER_apply_tophat
8               myPROTEOMICSWORKER_apply_tophat =
9               new PROTEOMICSWORKER_apply_tophat(
10                    "PROTEOMICSWORKER_apply_tophat.java", "application__tophat",
11                    "", false, "", "", args[0]);
12
13        % run main method
14
15        myPROTEOMICSWORKER_apply_tophat.do_work();
16     }
17
18     ...
19
20     // ----------------------------------------------------------
21     public void do_work() {
22
23     ...
24
25     params = get_next_task();
26
27     if ( (params != null) && (!params.equals(("")))  ) {
28         task_handled = true;
```

```

    // open task found
    change_state("Processing Task_ID: " + cur_task_ID);

    // process task

    String source_file = params[1];
    String target_file = source_file + ".filtered";
    String tophat_param = params[2];
    String target_dir = params[4];

    String cmd = "/home/cocktail/conrad/work/phd/PROTEOMICS_WORKER/
                    PROTEOMICSWORKER_apply_tophat/ UseTopHat " +
                    source_file + " " + tophat_param;

    exec_cmd(cmd);

    ...


    // Move file to new directory
    res = f_source.renameTo(new File(target_dir_handle,

    ...
```

6.2.6 Allowing for Easy Creation and Integration of Grid Applications on the Example of Analyzing Molecular Dynamics Simulations

Background

The goal of this project was to build a pipeline for dimensionality reduction and analysis of molecular dynamics simulation data. This is a three step process:

Import data into the system: To enable access to the data it must be copied to a temporary directory of this project accessible by the server. Subsequently, the datasets can be selected for import. This includes extraction of meta-data (such as structure of the simulated molecule) and conversion into the “trr” (Gromacs trajectory format) format, which is used internally. After successful conversion the transformed data is moved to the project directory and a description, metadata and an ID is inserted into the database. The ID is necessary to access the data in further analyses.

Transform data: To prepare the data for analysis it must be converted to an internal binary format that can be read by the analysis algorithms. Because an analysis only needs parts of the full dataset this step is not included in the import process.

Analyze data: The analysis of the transformed data is performed and the results written to the database.

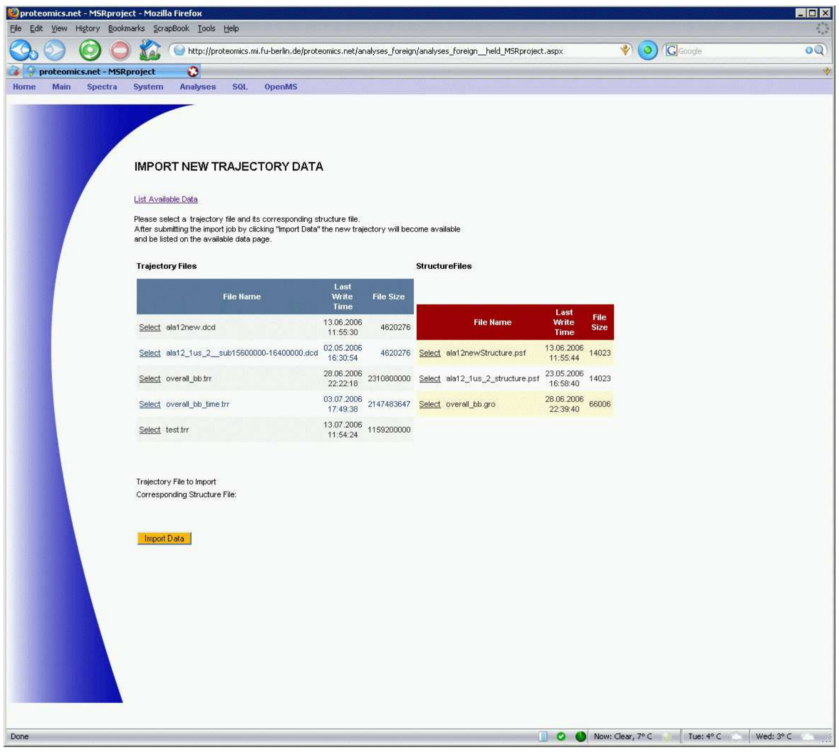
Since the data is usually quite large (several Gigabytes) it makes perfectly sense to process many datasets in parallel.

Approach

Based on the pipeline described above we implemented a graphical user interface (GUI) and the import, transform and analyses workers which are described below.

Graphical user interface Based on the *proteomics.net* design principles we integrated the GUI into the framework. It allows to import datasets (see figure 6.4(a)), transformation of data (figure 6.4(b) and 6.5(a)) and visualization of results (see Figure 6.5(b)).

QAD Grid workers Three workers were needed for the tasks of conversion, transformation and the actual analysis and reflect the full spectrum of possible applications (see section 5.4.1): The **conversion worker** is a *wrapper worker* that calls an external tool. The **transformation worker** is a *full worker* that implements the complete algorithm within the QAD Grid worker framework. Finally, the **analysis worker** is a *proxy worker* that calls methods from an external library to perform the analysis.


proteomics.net - MSRproject - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/analyses_foreign/analyses_foreign__held_MSRproject.aspx
Google
proteomics.net - MSRproject
Home Main Spectra System Analyses SQL OpenMS
Done
Now: Clear, 7° C
Tue: 4° C
Wed: 3° C

IMPORT NEW TRAJECTORY DATA

List Available Data

Please select a trajectory file and its corresponding structure file.
After submitting the import job by clicking "Import Data" the new trajectory will become available and be listed on the available data page.

Trajectory Files

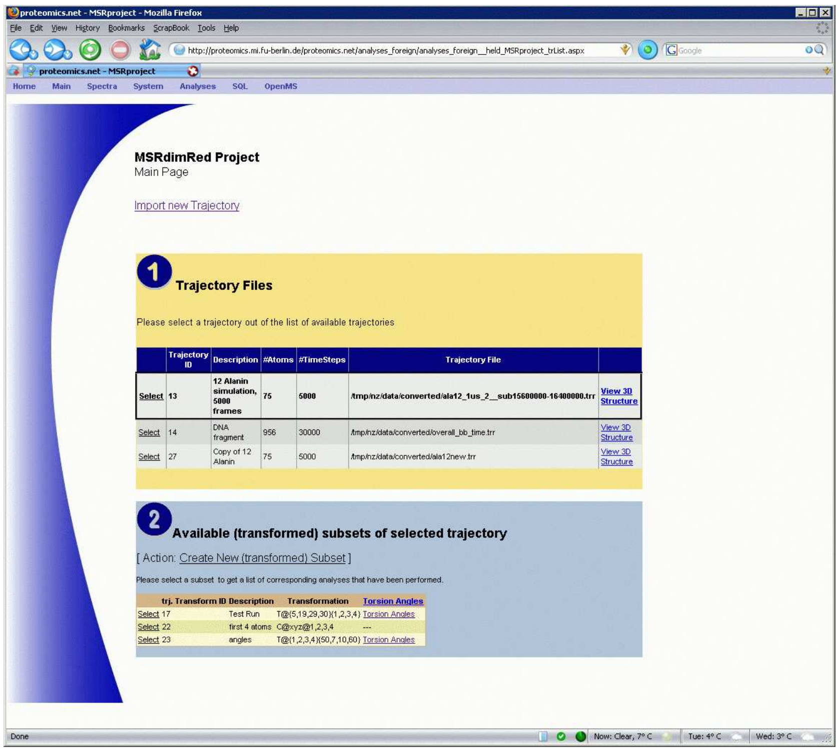
	File Name	Last Write Time	File Size
Select	ala12new.dcd	13.06.2006 11:55:30	4620276
Select	ala12_1us_2__sub15600000-16400000.dcd	02.05.2006 16:30:54	4620276
Select	overall_bb.trr	28.06.2006 22:22:18	2310800000
Select	overall_bb_time.trr	03.07.2006 17:49:38	2147483647
Select	test.trr	13.07.2006 11:54:24	1159200000

StructureFiles

	File Name	Last Write Time	File Size
Select	ala12newStructure.psf	13.06.2006 11:55:44	14023
Select	ala12_1us_2_structure.psf	23.05.2006 16:58:40	14023
Select	overall_bb.gro	28.06.2006 22:39:40	66006

Trajectory File to Import
Corresponding Structure File:

Import Data


proteomics.net - MSRproject - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/analyses_foreign/analyses_foreign__held_MSRproject_trList.aspx
Google
proteomics.net - MSRproject
Home Main Spectra System Analyses SQL OpenMS
Done
Now: Clear, 7° C
Tue: 4° C
Wed: 3° C

MSRdimRed Project
Main Page

Import new Trajectory

1 **Trajectory Files**

Please select a trajectory out of the list of available trajectories

	Trajectory ID	Description	#Atoms	#TimeSteps	Trajectory File	
Select	**13**	**12 Alanin simulation, 5000 frames**	**75**	**5000**	**/tmp/nz/data/converted/ala12_1us_2__sub15600000-16400000.trr**	**View 3D Structure**
Select	14	DNA fragment	956	30000	/tmp/nz/data/converted/overall_bb_time.trr	View 3D Structure
Select	27	Copy of 12 Alanin	75	5000	/tmp/nz/data/converted/ala12new.trr	View 3D Structure

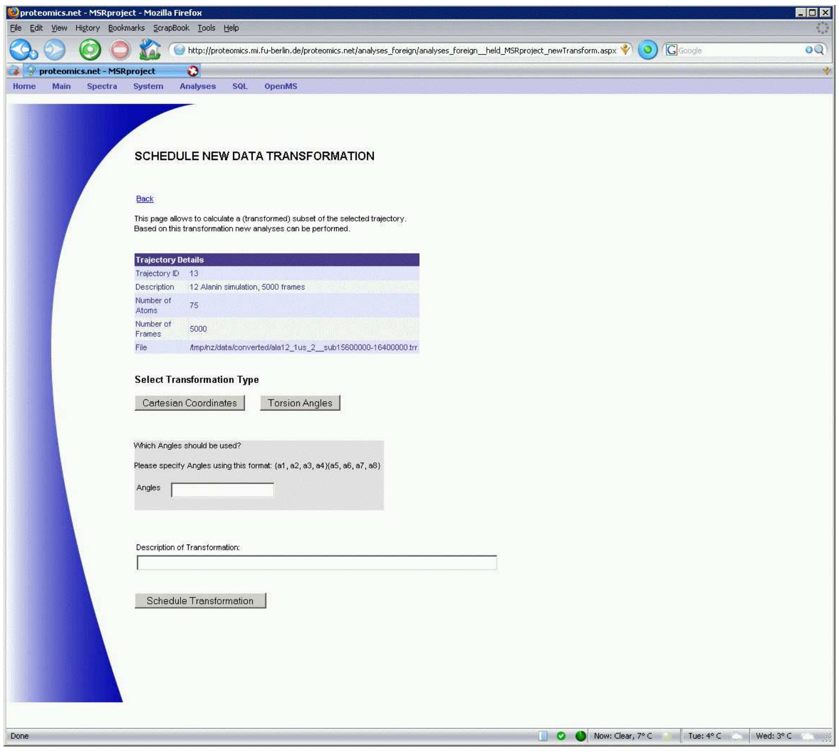
2 **Available (transformed) subsets of selected trajectory**

[Action: Create New (transformed) Subset]

Please select a subset to get a list of corresponding analyses that have been performed.

	trj. Transform ID	Description	Transformation	Torsion Angles
Select	17	Test Run	T@{5,19,29,30}{1,2,3,4}	Torsion Angles
Select	22	first 4 atoms	C@xyz@1,2,3,4	---
Select	23	angles	T@{1,2,3,4}{50,7,10,60}	Torsion Angles

Figure 6.2.4: Screenshot of the proteomics.net GUI for MD simulation analysis: Top: Import of datasets. Bottom: Available datasets and links to the results.


proteomics.net - MSRproject - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/analyses_foreign/analyses_foreign__held_MSRproject_newTransform.aspx
Google
proteomics.net - MSRproject
Home Main Spectra System Analyses SQL OpenMS

SCHEDULE NEW DATA TRANSFORMATION

Back

This page allows to calculate a (transformed) subset of the selected trajectory.
Based on this transformation new analyses can be performed.

Trajectory Details	
Trajectory ID	13
Description	12 Alanin simulation, 5000 frames
Number of Atoms	75
Number of Frames	5000
File	/tmp/nz/data/converted/ala12_1us_2__sub15600000-16400000.trr

Select Transformation Type

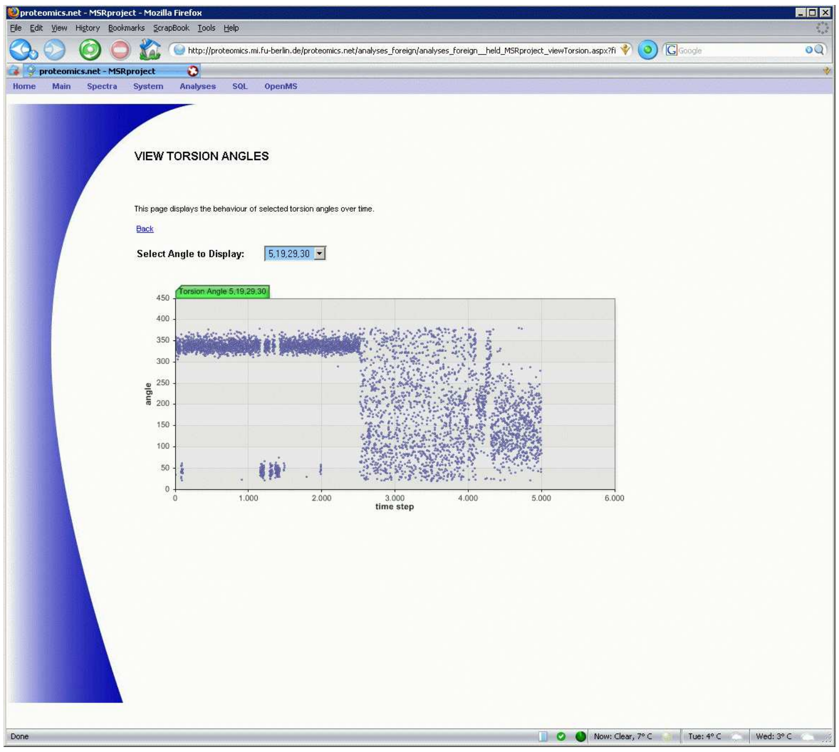
Which Angles should be used?

Please specify Angles using this format: {a1, a2, a3, a4}{a5, a6, a7, a8}

Angles

Description of Transformation:

Schedule Transformation
Done
Now: Clear, 7° C
Tue: 4° C
Wed: 3° C


proteomics.net - MSRproject - Mozilla Firefox
File Edit View History Bookmarks ScrapBook Tools Help
http://proteomics.mi.fu-berlin.de/proteomics.net/analyses_foreign/analyses_foreign__held_MSRproject_viewTorsion.aspx?fi
Google
proteomics.net - MSRproject
Home Main Spectra System Analyses SQL OpenMS

VIEW TORSION ANGLES

This page displays the behaviour of selected torsion angles over time.

Back

Select Angle to Display: 5,19,29,30

Torsion Angle 5,19,29,30
450
400
350
300
250
200
150
100
50
0
angle
0
1.000
2.000
3.000
4.000
5.000
6.000
time step
Done
Now: Clear, 7° C
Tue: 4° C
Wed: 3° C

Figure 6.2.5: Screenshot of the proteomics.net GUI for MD simulation analysis: Top: transformation of data, bottom: visualization of results.

Chapter 7

Related Work

In this chapter we will give a brief overview of other projects related to this thesis. Since related algorithms and concepts have been already discussed in the relevant chapters we will focus here on whole pipelines or frameworks for the analysis of protein MS TOF data. These pipelines can be roughly categorized into three groups:

1. Collection of stand alone tools for data processing (e.g. peak picking, identification or quantitation)
2. Integrated software platforms that offer data processing tools and data management functionality, usually providing a graphical user interface that allows for the assembly and execution of workflows. Since these platforms run on a single machine they are not well suited for the analysis of very large datasets.
3. Software platforms that offer data processing, data management and support distributed computation of their algorithms. Opposed to the previous category these frameworks are also well suited to handle very large datasets.

Interestingly, the shift from development of stand-alone tools to integrated platforms has only become widespread since increase in data volume became an issue. Still, most of these systems address only a part of the pipeline described in this thesis. High-throughput laboratories such as the Seattle Proteome Center (Trans-Proteomics Pipeline[1], TPP (Kiebel et al., 2006)), the Institute of Molecular Systems Biology at ETH Zürich (Superhirn, (Mueller et al., 2007)) or the Institute of Biomedical Engineering at Imperial College London (ProteomeGRID, (Dowsey et al., 2004)) have developed significant platforms with similar functionality. However, none of them provides the full range of the discovery process lifecycle including distributed computing support as our platform does.

Other software packages such as mzMine[2] by Turku Centre for Biotechnology (Katajamaa et al., 2006), OpenMS[3] by Freie Universität Berlin (Kohlbacher et al., 2007) or XCMS[4] by Scripps Center for Mass Spectrometry (Smith et al., 2006) are not complete platforms (yet) since there is no graphical user interface, workflow or data management functionality.

[1] `http://tools.proteomecenter.org/wiki/index.php?title=Software:TPP`
[2] `http://mzmine.sourceforge.net/`
[3] `http://www.openms.de`
[4] `http://metlin.scripps.edu/download/`

There are also platforms such as GenePattern[5] (Reich et al., 2006) that extend other systems (PEPPeR (Jaffe et al., 2006)) by data management functionality and further analysis algorithms.

Besides the academic efforts there are of course also commercial solutions either by instrument vendors (e.g. ClinProt Tools by Bruker Daltronics, Protein Expression Systems by Waters, ProteinChip by BioRad, ProteinPilot by Applied Biosystems) or other proteomics companies (e.g. CellCarta by caprion proteomics, ProteomeIQ by proteome systems, or Progenesis by Nonlinear Dynamics). Although a comprehensive comparison of these products is not possible, in conclusion it can be said that all of them share the common goal of biomarker detection from MS data.

[5]`http://www.broad.mit.edu/cancer/software/genepattern/desc/proteomics.html`

Chapter 8

Conclusion and Future Directions

8.1 Conclusion

In this thesis we have described a new pipeline for preprocessing, processing and analysis of Time-Of-Flight Mass Spectrometry proteomics data. We have shown the application of this web-based framework in the area of clinical diagnostics to detect molecular patterns (fingerprints) of diseases and used these to classify unknown datasets (spectra). Newly developed algorithms allow the detection of very small signals that presumably represent very low abundant molecules, such as hormones. Fingerprints including these very small signals can be much more sensitive compared to standard approaches that do not detect them.

The datasets are typically very large (several Gigabytes per patient). To enable handling of these large datasets we have also introduced a new distributed computing platform that allows creation of heterogeneous ad-hoc Grids. To demonstrate the universality of this framework we have shown in a case study the integration of a Playstation 3 to perform data preprocessing tasks.

In the current stage of this work it is possible to reliably detect multi-component fingerprints for a given disease. The components of a fingerprint represent particular molecular species (peptides or proteins) contained in some body fluid (e.g. blood serum) that significantly differentiate in concentration between two patient groups, such as healthy vs. diseased. These molecules - once identified - can be used as so called *biomarkers*. A biomarker is defined as a molecular, biological or physical characteristic (e.g. DNA sequences, antibodies or - as in our case - proteins) that indicates a specific physiologic state which can be directly linked to the clinical manifestations and outcome of a particular disease.

After identification of a disease specific set of biomarkers subsequent studies can be designed to eventually find drugs that can target these biomarkers and hopefully open up new steps for actually curing this disease.

However, until this vision becomes reality there are some open problems that need further investigation to allow for a deeper understanding and thus reliable and comprehensible mass-spectrometry based clinical diagnostics. These open problems include:

1. Mass spectrometry is a largely *qualitative technique*. That is, the actual serum concentration of discriminating molecules is not known and can only be stated as relative differences. However, it could be (in principle) inferred by comparisons to known concentrations. This is further complicated by the fact that relationship between peak area (or height) and molecule abundance is not linear and could be very complex.

2. Analysis of current literature has shown that *discriminating peaks* for a particular disease identified by different investigators *are mostly different.*

3. Due to the lack of standards in pre-analysis preparation of the biological samples, data acquired by different laboratories even from the same sample is often quite different. This *problem in reproducibility* of results extremely hinders validation.

4. Since we are dealing with biological samples protease activity can have a large impact on the results. Thus, it is *often not clear whether discriminatory peaks originate by the action of proteases* after the blood is drawn.

Another important issue for the detection of reliable biomarkers is the biological material used to perform the studies. In this thesis we mainly used material from voluntary blood donors or from patients from one hospital that suffered from a particular cancer. These patient groups are usually of small to medium size since - fortunately - there are not that many people in a hospital that suffer from the same type of disease under investigation.

The problem of a small patient group is statistical significance of the results obtained by analyzing this group. Therefore, for a study to deliver reliable results the number of analyzed patients needs to be sufficiently large. *Biobanks* are (usually very large) collections of biological material such as cells, tissue or body fluids (e.g. blood or urine) including meta-data of the donors, for example age, sex, blood parameters or informations about existing diseases. As explained in more detail below, these biobanks can provide enough biological material allowing to study even rare diseases.

8.2 From Biobanks to Biomarkers

Some European countries consider basically any repository of biological material as a biobank, whereas in other countries biobanks are seen as a research infrastructure (EU, 2003). There is not *a typical* biobank; they differ based on the type of samples that are stored and the scientific domain in which they are collected (Cambon-Thomsen, 2004). There are at least three major types of biobanks:

- *Population based biobanks* aim to study the development of common, complex diseases over time. This type seems to dominate the public perception of biobanks and their associated scientific, ethical, legal and social issues (Lipworth, 2005; Joly and Knoppers, 2006). Examples for this type of biobanks are the first initiative, the Icelandic BioGenetic Project (Hodgson, 1998; Palsson and Rabinow, 1999) and follow-up projects e.g. in the UK (1998), Norway (1999), Japan (2003) or the USA (2004) (Maschke, 2005). Beyond these large projects, there have also been many smaller biobanks as well as existing epidemiological studies rebranding themselves as "biobanks" (Gibbons et al., 2007).

- *Corporate-held biobanks* involve the collection of tissue samples and clinical data by pharmaceutical companies and clinical research organizations from clinical trial subjects. Opposed to the population-based biobanks, the corporate biobanks are not as well documented or scrutinized by ethicists, lawyers or social scientists (Corrigan and Williams-Jones, 2006). This can be partly explained by the fact that details about these biobanks are commercially sensitive and therefore mostly kept secret. There is evidence that pharmaceutical companies such as Novartis, Roche and Pfizer have been routinely collecting biological samples from clinical trials and have created large repositories of tissues with assigned patient information (Lewis, 2004).

- *Disease specific biobanks* which are established by disease advocacy organizations with the aim of producing therapies for people with rare genetic conditions. One of the earliest examples is PXE International (Terry et al., 2007) collecting tissues and patient information of people affected by the rare genetic disorder of pseudoxanthona elasticum (Zarbock et al., 2007).

Biobanks as a basis for better drug development

Biobanks have been part of what has been called the *biotechnology revolution* (Nightingale and Martin, 2004) - a view that was shared by governments, academics and industry. In short, it was believed that significant benefit would come from genomics and biotechnology (by using these biobanks) for drug development, healthcare and the economy in general. This was one of the main reasons for governmental support in the creation of large population-based biobanks.

Thus, the central expectation of biobank research is that it will enhance diagnosis, prevention and treatment of diseases, leading to an improvement in the health of the general population and in particular subgroups. This hope is based on the main assumption that analysis of biobanks will lead to better understanding of diseases which is usually coupled to the identification of *biomarkers* for a particular disease. There is no doubt that the establishment of cooperative human tissue banks or research networks can greatly facilitate the large-scale validation of biomarkers.

8.3 From Biomarkers to Bioprints: Enabling Information-Based Medicine

The concept of personalized medicine embodies the belief that a drug is not simply effective or ineffective. Rather, it is likely to be more effective in some people and less effective or even harmful in others. Thus, personalized medicine can improve the potential for successful, sophisticated evaluation of the balance of risk and benefit. The promise of genomics and proteomics is largely built on the theory that these technologies will enable us to judge these risk and benefit consideration using much smaller, focused groups of patients than before. As a consequence, the concept of single-source biomarkers (e.g. *just* a genomic SNP or *just* proteomics peptide modification) used to make these judgements, should be replaced by multi-source (or multi-*dimensional*) biomarkers that combine informations from different sources in a reasonable way.

What we call a *bioprint* is the multi-dimensional extension of the biomarker concept. Using many different data sources (-omics) and different analyzing techniques, bioprints can be created that not only rely on a single change but mathematically and biologically meaningful combine many different relevant elements. These potentially complex models require not only sophisticated data-mining algorithms for the analysis but also large amounts of computing power.

The ultimate goal should be to find disease specific bioprints using data from all available kinds of sources and examine the relationships between clinical symptoms and genetic, biochemical, immunological and cellular biomarkers. This and perhaps only this would facilitate an area of *personalized medicine*: "Although they are certainly related, the only way to really achieve personalized medicine is to be able to create an information base that lets you say what is it about an individual that I need to know in order to define what the right treatment is." (Carol Kovac, General Manager Healthcare and Life Sciences at IBM interviewed by (McDonald, 2006).)

The Next Steps

We feel that the there are four main steps that have to be taken in order to come closer to the aim of providing such an information base:

1. **Evaluate available data:** Collect, classify and analyze publicly available biobank data and data available in "unofficial biobanks" such as hospitals or institutes.

2. **Consolidate compatible data on a disease level:** Based on the analysis in the previous step, data clusters are created that are related to the focused cancer types. Subsequently, sub-clusters are extracted that contain compatible data, that is, data that can reasonably be compared with respect to biology.

3. **Data Mining:** Develop algorithms to analyze and mine this data.

4. **Web-based community access:** Build and provide an "out-of-the-box" web-based platform for researchers across disciplines and medical staff to access this data and perform individual analyses.

The ultimate goal: Value for Healthcare

By developing new ways to identify major diseases at the molecular level and provide appropriate (personalized) diagnosis and therapeutics. This means shorter duration of therapy, increased healing chances and therefore increases life quality.

Appendix A

Implementation Details

The project was implemented using the following programming languages / tools:

Worker: Matlab, Java and C++ (32.808 lines in total)

Database: Microsoft SQl Server 2005 (64 tables)

Server: .NET 2.0, C# and Visual Basic (26.735 lines in total)

Webserver: Microsoft Internet Information Server

Appendix B

Curriculum Vitae

For data privacy reasons, this online version does not contain the CV.

References

Aebersold, R. and Mann, M. (2003). Mass spectrometry-based proteomics., *Nature* **422**(6928): 198–207.

Aittokallio, T., Nevalainen, O., Ojala, P. and Nevalainen, T. J. (2001). Automated detection of differentially expressed fragments in mRNA differential display, *Electrophoresis* **22**(10): 1935–1045.

Aldous, D. (1983). *Exchangeability and related topics*, Vol. 1117 of *Lecture Notes in Math - Ecole d'ete de probabilites de Saint-Flour*, Springer, Berlin.

Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., Meder, S. and Tuecke, S. (September 2002). Gridftp protocol specification, GGF GridFTP Working Group Document.

America, A., Cordewener, J., van Geffen, M., Lommen, A., Vissers, J., Bino, R. and Hall, R. (2006). Alignment and statistical difference analysis of complex peptide data sets generated by multidimensional lc-ms, *Proteomics* **6**: 641–653.

Anderson, D. P. (2004). Boinc: a system for public-resource computing and storage, pp. 4–10.

Anderson, T. W. and Darling, D. A. (1952). Asymptotic theory of certain "goodness of fit" criteria based on stochastic processes, *The Annals of Mathematical Statistics* **23**(2): 193–212.

Anthony, D. (1996). A review of statistical methods in the journal of advanced nursing., *J Adv Nurs* **24**(5): 1089–1094.

Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K. and Mitchell, J. S. B. (1991). An efficiently computable metric for comparing polygonal shapes, *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(3): 209–216.

Asadzadeh, P., Buyya, R., Kei, C., Nayar, D. and Venugopal, S. (2006). *Global Grids and Software Toolkits: A Study of Four Grid Middleware Technologies*, Wiley Series on Parallel and Distributed Computing, Wiley Press, New Jersey, USA, chapter 22, pp. 431–459.

Baggerly, K. A., Morris, J. S., Wang, J., Gold, D., Xiao, L.-C. and Coombes, K. R. (2003). A comprehensive approach to the analysis of matrix-assisted laser desorption/ionization-time of flight proteomics spectra from serum samples., *Proteomics* **3**(9): 1667–1672.

Bakan, D. (1970). *The test of significance in psychological research*, Butterworth, NY.

Banks, R. E., Stanley, A. J., Cairns, D. A., Barrett, J. H., Clarke, P., Thompson, D. and Selby, P. J. (2005). Influences of blood sample processing on low-molecular-weight proteome identified by surface-enhanced laser desorption/ionization mass spectrometry., *Clin Chem* **51**(9): 1637–1649.

Barak, A. and La'adan, O. (1998). The MOSIX multicomputer operating system for high performance cluster computing, *Future Generation Computer Systems* **13**(4–5): 361–372.

Baumann, J., Hohl, F., Radouniklis, N., Rothermel, K. and Strasser, M. (1997). Communication concepts for mobile agent systems, *MA '97: Proceedings of the First International Workshop on Mobile Agents*, Springer-Verlag, London, UK, pp. 123–135.

Baumann, S., Ceglarek, U., Fiedler, G. M., Lembcke, J., Leichtle, A. and Thiery, J. (2005). Standardized approach to proteome profiling of human serum based on magnetic bead separation and matrix-assisted laser desorption/ionization time-of-flight mass spectrometry., *Clin Chem* **51**(6): 973–980.

Becker, S., Cazares, L. H., Watson, P., Lynch, H., Semmes, O. J., Drake, R. R. and Laronga, C. (2004). Surfaced-enhanced laser desorption/ionization time-of-flight (SELDI-TOF) differentiation of serum protein profiles of BRCA-1 and sporadic breast cancer., *Ann Surg Oncol* **11**(10): 907–914.

Benoit, E. and Foulloy, L. (2003). Towards fuzzy nominal scales, *Measurement* **34**(1): 49–55.

Bera, A. and Carlos, M. (1980). Efficient tests for normality, homoscedasticity and serial independence of regression residuals, *Economics Letters* **6**(3): 255259.

Besbeas, P., Feis, I. D. and Sapatinas, T. (2004). A comparative simulation study of wavelet shrinkage estimators for poisson counts, *International Statistical Review* **72**: 209–237.

Beyer, K., Goldstein, J., Ramakrishnan, R. and Shaft, U. (1999). When is "nearest neighbor" meaningful?, *Lecture Notes in Computer Science, Proceedings of the 7th International Conference on Database Theory* **1540**: 217–235.

Bezdek, J. (1981). *Pattern recognition with fuzzy objective function algorithms.*, Plenum Press, New York.

Bezdek, J., Hathaway, R., Sabin, M. and Tucker, W. (1987). Convergence theory for fuzzy c-mens: counterexamples and repairs., *IEEE Trans. Systems.* **17**: 873–877.

Blackwell, D. and MacQueen, J. (1973). Ferguson distributions via polya urn schemes, *The Annals of Statistics* **1**: 353–355.

Boguski, M. and McIntosh, M. (2003). Biomedical informatics for proteomics, *Nature* **422**(6928): 233–237.

Breen, E. J., Hopwood, F. G., Williams, K. L. and Wilkins, M. R. (2000). Automatic poisson peak harvesting for high throughput protein identification, *Electrophoresis* **21**(11): 2243 – 2251.

Breiman, L. (2001). Random forests, *Machine Learning* **45**(1): 5–32.

Brix, T. H., Hansen, P. S., Kyvik, K. O. and Hegeds, L. (2004). Aggregation of thyroid autoantibodies in first-degree relatives of patients with autoimmune thyroid disease is mainly due to genes: a twin study., *Clin Endocrinol (Oxf)* **60**(3): 329–334.

Brown, L. M., Helmke, S. M., Hunsucker, S. W., Netea-Maier, R. T., Chiang, S. A., Heinz, D. E., Shroyer, K. R., Duncan, M. W. and Haugen, B. R. (2006). Quantitative and qualitative differences in protein expression between papillary thyroid carcinoma and normal thyroid tissue., *Mol Carcinog* **45**(8): 613–626.

Bundesgesundheitsblatt-Gesundheitsforschung-Gesundheitsschutz (2000). Guidelines for manufacturing of blood and blood components and for the use of blood products (hemotherapy). 43:555-589.

Butcher, D. N., Gregory, W. M., Gunter, P. A., Masters, J. R. and Parkinson, M. C. (1985). The biological and clinical significance of hcg-containing cells in seminoma., *Br J Cancer* **51**(4): 473–478.

Cambon-Thomsen, A. (2004). The social and ethical issues of post-genomic human biobanks., *Nat Rev Genet* **5**(11): 866–873.

Campbell, N. R. (1920). *Physics: The Elements*, n/a, Cambridge.

Carl, A., Laurberg, P., Knudsen, N., Perrild, H., Ovesen, L., Rasmussen, L. B., Jorgensen, T. and Pedersen, I. B. (2006). Thyroid peroxidase and thyroglobulin auto-antibodies in patients with newly diagnosed overt hypothyroidism., *Autoimmunity* **39**(6): 497–503.

Carriero, N. and Gelernter, D. (1986). The s/net's linda kernel, *ACM Trans. Comput. Syst.* **4**(2): 110–129.

Caws, P. (1959). *Measurement: Definitions and Theories*, John Wiley & Sons, New York.

CERN (2007). The grid cafe, `http://gridcafe.web.cern.ch`, accessed October 11, 2007.

Chase, J., Gallatin, A. and Yocum, K. (2001). End system optimizations for high-speed TCP, *IEEE Communications Magazine* **39**(4): 68–74.

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

Cohen, J. (1969). *Statistical power analysis for the behavioral sciences*, Academic Press, New York.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*, 2nd edn, Hillsdale, NJ.

Cohen, J. (1990). Things i have learned (so far), *American Psychologist* **45**: 1304–1312.

Coombes, K. R., Tsavachidis, S., Morris, J. S., Baggerly, K. A., Hung, M.-C. and Kuerer, H. M. (2005). Improved peak detection and quantification of mass spectrometry data acquired from surface-enhanced laser desorption and ionization by denoising spectra with the undecimated discrete wavelet transform., *Proteomics* **5**(16): 4107–4117.

Corrigan, O. P. and Williams-Jones, B. (2006). Pharmacogenetics: the bioethical problem of dna investment banking., *Stud Hist Philos Biol Biomed Sci* **37**(3): 550–565.

Covey, T. R., Lee, E. D. and Henion, J. D. (1986). High-speed liquid chromatography/tandem mass spectrometry for the determination of drugs in biological samples., *Anal Chem* **58**(12): 2453–2460.

Cravatt, B. F., Simon, G. M. and Yates, J. R. (2007). The biological impact of mass-spectrometry-based proteomics., *Nature* **450**(7172): 991–1000.

Daemen, J. and Rijmen, V. (1999). Aes proposal: Rijndael, Technical Specification.

Daemen, J. and Rijmen, V. (2002). *The design of Rijndael: AES—the advanced encryption standard*, Springer-Verlag.

Danial, N. N., Gramm, C. F., Scorrano, L., Zhang, C.-Y., Krauss, S., Ranger, A. M., Datta, S. R., Greenberg, M. E., Licklider, L. J., Lowell, B. B., Gygi, S. P. and Korsmeyer, S. J. (2003). Bad and glucokinase reside in a mitochondrial complex that integrates glycolysis and apoptosis., *Nature* **424**(6951): 952–956.

de Berg, M., van Krevald, M., Overmars, M. and Schwarzkopf, O. (2000). *Computational Geometry*, second revised edition edn, Springer-Verlag.

Deonia, S. C., Ruttb, B. K., Parrentb, A. G. and Peters, T. M. (2007). Segmentation of thalamic nuclei using a modified k-means clustering algorithm and high-resolution quantitative magnetic resonance imaging at 1.5 t, *NeuroImage* **34**(1): 117–126.

Devlin, S., Gnanadesikan, R. and Kettenring, J. (1975). Robust estimation and outlier detection with correlation coefficients, *Biometrika* **61**: 531–545.

Diamandis, E. (2004). How are we going to discover new cancer biomarkers? a proteomic approach for bladder cancer, *Clin Chem* **50**(5): 793–795.

Dong, M.-Q., Venable, J. D., Au, N., Xu, T., Park, S. K., Cociorva, D., Johnson, J. R., Dillin, A. and Yates, J. R. (2007). Quantitative mass spectrometry identifies insulin signaling targets in c. elegans., *Science* **317**(5838): 660–663.

Donoho, D. (1995). Denoising via soft thresholding, *IEEE Trans. Information Theory* **41**: 613–627.

Donoho, D. L. and Johnstone, I. M. (1995). Adapting to unknown smoothness via wavelet shrinkage, *Journal of the American Statistical Association* **90**(432): 1200–1224.

Douglis, F. (1990). *Transparent Process Migration in the Sprite Operating System*, PhD thesis, U.C. Berkeley. Report UCB/CSD 90/598.

Downard, K. M. and Morrissey, B. (2007). Fingerprinting a killer: surveillance of the influenza virus by mass spectrometry., *Analyst* **132**(7): 611–614.

Dowsey, A. W., Dunn, M. J. and Yang, G.-Z. (2004). Proteomegrid: towards a high-throughput proteomics pipeline through opportunistic cluster image computing for two-dimensional gel electrophoresis., *Proteomics* **4**(12): 3800–3812.

Eager, D. L., Lazowska, E. D. and Zahorjan, J. (1985). A comparison of receiver-initiated and sender-initiated adaptive load sharing (extended abstract), *SIGMETRICS '85: Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, ACM, New York, NY, USA, pp. 1–3.

Ebert, M. P. A., Korc, M., Malfertheiner, P. and Rcken, C. (2006). Advances, challenges, and limitations in serum-proteome-based cancer diagnosis., *J Proteome Res* **5**(1): 19–25.

Efron, B. (1979). Bootstrap methods: Another look at the jackknife, *The Annals of Statistics* **7**: 1–26.

Efron, B. and Tibshirani, R. J. (1994). *An Introduction to the Bootstrap*, Chapman & Hall/CRC.

Elfwing, R., Paulsson, U. and Lundberg, L. (2002). Performance of soap in web service environment compared to corba, *APSEC '02: Proceedings of the Ninth Asia-Pacific Software Engineering Conference*, IEEE Computer Society, Washington, DC, USA, p. 84.

Emmerich, W. and Kaveh, N. (2002). Component technologies: Java beans, com, corba, rmi, ejb and the corba component model, *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, ACM Press, New York, NY, USA, pp. 691–692.

Engum, A., Bjro, T., Mykletun, A. and Dahl, A. (2005). Thyroid autoimmunity, depression and anxiety; are there any connections? an epidemiological study of a large population., *Journal of Psychosomatic Research* **59**: 263–268.

EU (2003). Eu workshop in fp5: Biobanks for health. optimising the use of european biobanks and health registries for research relevant to public health and for combating disease. report and recommendations., Oslo.

Fan, X. and Wang, L. (1995). How comparable are the jackknife and bootstrap results: An investigation for a case of canonical correlation analysis., *Proceedings of the annual meeting of the American Educational Research Association.*

Fasano, G. and Franceschini, A. (1987). A multidimensional version of the kolmogorov-smirnov test, *Monthly Notices of the Royal Astronomical Society* **225**: 155–170.

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

Fenn, J., Mann, M., Meng, C., Wong, S. and Whitehouse, C. (1990). Electrospray ionization-principles and practice, *Mass Spectrometry Reviews* **9**(1): 37–70.

Ferguson, T. S. (1973). A bayesian analysis of some nonparametric problems, *The Annals of Statistics* **1**: 209–230.

Finkelstein, L. and Leaning, M. (1984). A review of the fundamental concepts of measurement, *Measurement* **2**(1): 25–34.

Finnigan (2007). Maldi mass analysis, `http://www.biotech.iastate.edu/facilities/protein/maldi.html`, accessed August 5, 2007.

Fisher, R. (1922). On the mathematical foundations of theoretical statistics, *Philosophical Transactions of the Royal Society of London (A)* **222**: 309–368.

Fisher, R. A. (1959). Mathematical probability in the natural sciences, *Technometrics* **1**: 21–29.

Fisher, R. and Tippett, L. (1928). Limiting forms of the frequency distribution of the largest or smallest member of a sample., *Cambridge Phil. Soc.* **24**.

Foster, I. and Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit, *The International Journal of Supercomputer Applications and High Performance Computing* **11**(2): 115–128.

Fowler, M., Rice, D. and Foemmel, M. (2003). *Patterns of Enterprise Application Architecture*, Addison-Wesley Professional.

Freund, Y. and Mason, L. (1999). The alternating decision tree learning algorithm,, *Proc. 16th International Conf. on Machine Learning*, Morgan Kaufmann, San Francisco, CA, pp. 124–133.

Fung, E. T. and Enderwick, C. (2002). ProteinChip clinical proteomics: computational challenges and solutions., *Biotechniques* **Suppl**: 34–8, 40–1.

Gage, N. (1978). *The scientific basis of the art of teaching*, Teachers College Press, New York.

Gao, H., Xue, P., Lin, W. and Hou, C. (2003). Disjoint set data structure for morphological area operators, *Proceedings of the 2003 Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing*, Vol. 2, pp. 773–777.

Geisler, E. (2000). *The Metrics of Science and Technology*, Quorum Books, Westport, CT.

Gentzsch, W. (2001). Sun grid engine: towards creating a compute power grid, pp. 35–36.

Gibbons, S., Kaye, J. and Smart, A. (2007). Governing genetic databases: Challenges facing research regulation and practice, *JOURNAL OF LAW AND SOCIETY* **34**(2): 163–189.

Gill, A. (1962). *Introduction to the Theory of Finite-state Machines*, McGraw-Hill, New York.

Gillette, M. A., Mani, D. R. and Carr, S. A. (2005). Place of pattern in proteomic biomarker discovery., *J Proteome Res* **4**(4): 1143–1154.

Giustarini, E., Pinchera, A., Fierabracci, P., Roncella, M., Fustaino, L., Mammoli, C. and Giani, C. (2006). Thyroid autoimmunity in patients with malignant and benign breast diseases before surgery., *Eur J Endocrinol* **154**(5): 645–649.

Glass, G., McGaw, B. and Smith, M. (1981). *Meta-Analysis in Social Research*, Sage Publications.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Gómez-Lopera, J. F., Martínez-Aroza, J., Robles-Pérez, A. M. and Román-Roldán, R. (2000). An analysis of edge detection by using the jensen-shannon divergence, *J. Math. Imaging Vis.* **13**(1): 35–56.

Green, M. K., Johnston, M. V. and Larsen, B. S. (1999). Mass accuracy and sequence requirements for protein database searching, *Analytical Biochemistry* **275**: 39–46.

GridInfoware (2008). What is proteomics?, `http://www.gridcomputing.com`, accessed October 11, 2007.

Gröpl, C., Hildebrandt, A., Kohlbacher, O., Lange, E., Lövenich, S. and Sturm, M. (2005). OpenMS - Software for Mass Spectrometry, Poster presented at the MBI Workshop on Computational Proteomics and Mass Spectrometry 2005, Ohio State University.

Guiasu, S. and Shenitzer, A. (1985). The principle of maximum entropy, *The Mathematical Intelligencer* **7**(1): 42–48.

Guilhaus, M. (1995). Principles and instrumentation in time of flight mass spectrometry, *Journal of Mass Spectrometry* **30**: 1519–1532.

Gutekunst, R., Becker, W., Hehrmann, R., Olbricht, T. and Pfannenstiel, P. (1988). Ultrasonic diagnosis of the thyroid gland, *Dtsch Med Wochenschr* **113**: 1109–12.

Guyon, I., Weston, J., Barnhill, S. and Vapnik, V. (2002). Gene selection for cancer classification using support vector machines, *Machine Learning* **46**(1-3): 389–422.

Halligan, B., Ruotti, V., N, T. and Greene, A. (2005). A web-based tool for identifying peptides from sequence and mass tags deduced from de novo eptide sequencing by mass spectrometry, *Nucleic Acids Research* **33**: 376–381.

Hampel, R., Klberg, T., Klein, K., Jerichow, J. U., Pichmann, E. G., Clausen, V. and Schmidt, I. (1995). [goiter incidence in germany is greater than previously suspected], *Med Klin (Munich)* **90**(6): 324–329.

Hedges, L. and Olkin, I. (1985). *Statistical methods for meta-analysis*, Academic Press, San Diego, CA.

Henning, M. (2006). The rise and fall of corba, *Queue* **4**(5): 28–34.

Hernandez, P., Mller, M. and Appel, R. D. (2006). Automated protein identification by tandem mass spectrometry: issues and strategies., *Mass Spectrom Rev* **25**(2): 235–254.

Hinneburg, A., Aggarwal, C. C. and Keim, D. A. (2000). What is the nearest neighbor in high dimensional spaces?, pp. 506–515.

Hodgson, J. (1998). Iceland considers its genetic future., *Nat Biotechnol* **16**(10): 896–897.

Hoffman, B. and Diamandis, E. (2004). Recent advances in cancer biomarkers, *Clin Biochem* **37**(7): 503–504.

Hollowell, J. G., Staehling, N. W., Flanders, W. D., Hannon, W. H., Gunter, E. W., Spencer, C. A. and Braverman, L. E. (2002). Serum tsh, t(4), and thyroid antibodies in the united states population (1988 to 1994): National health and nutrition examination survey (nhanes iii)., *J Clin Endocrinol Metab* **87**(2): 489–499.

Honda, K., Hayashida, Y., Umaki, T., Okusaka, T., Kosuge, T., Kikuchi, S., Endo, M., Tsuchida, A., Aoki, T., Itoi, T., Moriyasu, F., Hirohashi, S. and Yamada, T. (2005). Possible detection of pancreatic cancer by plasma protein profiling., *Cancer Res* **65**(22): 10613–10622.

Hong, H., Dragan, Y., Epstein, J., Teitel, C., Chen, B., Xie, Q., Fang, H., Shi, L., Perkins, R. and Tong, W. (2005). Quality control and quality assessment of data from surface-enhanced laser desorption/ionization (seldi) time-of flight (tof) mass spectrometry (ms)., *BMC Bioinformatics* **6 Suppl 2**: S5.

Höppner, F., Klawonn, F., Kruse, R. and Runkler, T. (1999). *Fuzzy cluster analysis.*, John Wiley and Sons, New York.

Hsieh, Y. and Korfmacher, W. A. (2006). Increasing speed and throughput when using hplc-ms/ms systems for drug metabolism and pharmacokinetic screening., *Curr Drug Metab* **7**(5): 479–489.

Huang, H. H. and Grimshaw, A. S. (2006). The cost of transparency: Grid-based file access on the avaki data grid., *in* M. Guo, L. T. Yang, B. D. Martino, H. P. Zima, J. Dongarra and F. Tang (eds), *ISPA*, Vol. 4330 of *Lecture Notes in Computer Science*, Springer, pp. 642–659.

Huberty, C. and Morris, J. (1988). Multivariate analysis versus multiple univariate analyses, *Psychol. Bull.* **195**(3): 302–308.

Huddleston, J., Raghuram, R. and Gilani, S. F. (2006). *Beginning C# 2005 databases : from novice to professional*, Apress, Berkeley, CA.

Hugh P. Bivens, G. C. E. W. G. (2001). Grid workflow, Global Grid Forum.

HUPO (2005). What is proteomics?, `http://www.hupo.org/overview/background/proteomics.asp`, accessed October 11, 2007.

Hutchens, T. and Yip, T. (1993). New desorption strategies for the mass spectrometric analysis of macromolecules, *Rapid Commun Mass Spectrom* **7**: 576–580.

Huyghe, E., Muller, A., Mieusset, R., Bujan, L., Bachaud, J.-M., Chevreau, C., Plante, P. and Thonneau, P. (2007). Impact of diagnostic delay in testis cancer: results of a large population-based study., *Eur Urol* **52**(6): 1710–1716.

Huyghe, E., Plante, P. and Thonneau, P. F. (2007). Testicular cancer variations in time and space in europe., *Eur Urol* **51**(3): 621–628.

Ishwaran, H. and James, L. F. (2003). Generalized weighted chinese restaurant process for species sampling mixture models, *Statistica Sinica* **3**: 1211–1235.

J, B., U, B., G, R., I, B., WP, K. and PC, S. (1981). Volumetric analysis of thyroid lobes by real-time ultrasound (author's transl), *Dtsch Med Wochenschr* **106**: 1338–40.

J, J. E., McCormack, A. and Yates, J. (1994). An approach to correlate tandem mass spectral data of peptides with amino acid sequences in the protein database, *Journal of American Society of Mass Spectrometry* **5**: 976–989.

Jaffe, J. D., Mani, D. R., Leptos, K. C., Church, G. M., Gillette, M. A. and Carr, S. A. (2006). Pepper, a platform for experimental proteomic pattern recognition., *Mol Cell Proteomics* **5**(10): 1927–1941.

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

J.B. Tenenbaum, V. d. S. and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction, *Science* .

Jia, W., Zhang, H., He, X. and Wu, Q. (2006). Gaussian weighted histogram intersection for license plate classification, *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, IEEE Computer Society, Washington, DC, USA, pp. 574–577.

Johnson, S. (1967). Hierarchical clustering schemes, *Psychometrika* **32**(3): 241–254.

Joly, Y. and Knoppers, B. M. (2006). Pharmacogenomic data sample collection and storage: ethical issues and policy approaches., *Pharmacogenomics* **7**(2): 219–226.

Karanikas, G., Schuetz, M., Wahl, K., Paul, M., Kontur, S., Pietschmann, P., Kletter, K., Dudczak, R. and Willheim, M. (2005). Relation of anti-tpo autoantibody titre and t-lymphocyte cytokine production patterns in hashimoto's thyroiditis., *Clin Endocrinol (Oxf)* **63**(2): 191–196.

Kasagi, K., Kousaka, T., Higuchi, K., Iida, Y., Misaki, T., Alam, M. S., Miyamoto, S., Yamabe, H. and Konishi, J. (1996). Clinical significance of measurements of antithyroid antibodies in the diagnosis of hashimoto's thyroiditis: comparison with histological findings., *Thyroid* **6**(5): 445–450.

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

Keerthi, S., Shevade, S., Bhattacharyya, C. and Murthy, K. (1999). Improvements to platt's smo algorithm for svm classifier design.

Kesselman, C. and Foster, I. (1998). *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers.

Khan, S. M., Franke-Fayard, B., Mair, G. R., Lasonder, E., Janse, C. J., Mann, M. and Waters, A. P. (2005). Proteome analysis of separated male and female gametocytes reveals novel sex-specific plasmodium biology., *Cell* **121**(5): 675–687.

Kicman, A. T., Parkin, M. C. and Iles, R. K. (2007). An introduction to mass spectrometry based proteomics-detection and characterization of gonadotropins and related molecules., *Mol Cell Endocrinol* **260-262**: 212–227.

Kiebel, G. R., Auberry, K. J., Jaitly, N., Clark, D. A., Monroe, M. E., Peterson, E. S., Toli, N., Anderson, G. A. and Smith, R. D. (2006). Prism: a data management system for high-throughput proteomics., *Proteomics* **6**(6): 1783–1790.

Kim, S. B., Wang, Z. and Duran, C. M. (2006). A bayesian approach for the alignment of high-resolution nmr spectra, *Proceedings of the INFORMS Artificial Intelligence and Data Mining Workshop*, Pittsburgh , PA.

Koch, M., Mancini, L. V. and Parisi-Presicce, F. (2005). Graph-based specification of access control policies, *J. Comput. Syst. Sci.* **71**(1): 1–33.

Kohlbacher, O., Reinert, K., Grpl, C., Lange, E., Pfeifer, N., Schulz-Trieglaff, O. and Sturm, M. (2007). Topp–the openms proteomics pipeline., *Bioinformatics* **23**(2): e191–e197.

Koomen, J. M., Li, D., chun Xiao, L., Liu, T. C., Coombes, K. R., Abbruzzese, J. and Kobayashi, R. (2005). Direct tandem mass spectrometry reveals limitations in protein profiling experiments for plasma biomarker discovery., *J Proteome Res* **4**(3): 972–981.

Kozak, K. R., Su, F., Whitelegge, J. P., Faull, K., Reddy, S. and Farias-Eisner, R. (2005). Characterization of serum biomarkers for detection of early stage ovarian cancer., *Proteomics* **5**(17): 4589–4596.

Krantz, D., Luce, R. D., Suppes, P. and Tversky, A. (1971). *Foundations of Measurement: Additive and Polynomial Representations*, Vol. 1, Academic Press, New York.

Kratzsch, J., Fiedler, G. M., Leichtle, A., Brügel, M., Buchbinder, S., Otto, L., Sabri, O., Matthes, G. and Thiery, J. (2005). New reference intervals for thyrotropin and thyroid hormones based on National Academy of Clinical Biochemistry criteria and regular ultrasonography of the thyroid., *Clin Chem* **51**(8): 1480–1486.

Krause, K., Schierhorn, A., Sinz, A., Wissmann, J.-D., Beck-Sickinger, A. G., Paschke, R. and Fuhrer, D. (2006). Toward the application of proteomics to human thyroid tissue., *Thyroid* **16**(11): 1131–1143.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem, *Naval Research Logistic Quarterly* **2**: 83–97.

Kuijpens, J. L. P., Nyklctek, I., Louwman, M. W. J., Weetman, T. A. P., Pop, V. J. M. and Coebergh, J.-W. W. (2005). Hypothyroidism might be related to breast cancer in post-menopausal women., *Thyroid* **15**(11): 1253–1259.

Langley, P. (1994). Selection of relevant feature in machine learning, *Proceedings of the AAAI Fall Symposium on Relevance*, AAAI Press, New Orleans, p. 140144.

Lee, M. S. and Kerns, E. H. (1999). Lc/ms applications in drug development., *Mass Spectrom Rev* **18**(3-4): 187–279.

Lehmann, E. L. (1993). The fisher, neyman-pearson theories of testing hypotheses: One theory or two?, *Journal of the American Statistical Association* **88**(424): 1242–1249.

Lewis, G. (2004). *Tissue collection and the pharmaceutical industriy: corporate biobanks. In: Genetic Databases: Socio-ethical Issues in the Collection and Use of Dna, R Tutton and O Corrigan (Eds.)*, Routlege, London, UK.

Li, L., Tang, H., Wu, Z., Gong, J., Gruidl, M., Zou, J., Tockman, M. and Clark, R. A. (2004). Data mining techniques for cancer detection using serum proteomic profiling., *Artif Intell Med* **32**(2): 71–83.

Li, X., Li, J. and Yao, X. (2007). A wavelet-based data pre-processing analysis approach in mass spectrometry., *Comput Biol Med* **37**(4): 509–516.

Licklider, J. C. R. and Taylor, R. W. (1968). The computer as a communication device, *Sci. Technol.* .

Lilliefors, H. (1967). On the kolmogorov-smirnov test for normality with mean and variance unknown, *Journal of the American Statistical Association* .

Lin, J. (1991). Divergence measures based on the shannon entropy, *IEEE Trans. on Information Theory,* **37**(1): 145–151.

Lipworth, W. (2005). Navigating tissue banking regulation: conceptual frameworks for researchers, administrators, regulators and policy-makers., *J Law Med* **13**(2): 245–255.

Little, D. P., Speir, J. P., Senko, M. W., O'Connor, P. B. and McLafferty, F. W. (1994). Infrared multiphoton dissociation of large multiply charged ions for biomolecule sequencing., *Anal Chem* **66**(18): 2809–2815.

Liu, B.-F., Sera, Y., Matsubara, N., Otsuka, K. and Terabe, S. (2003). Signal denoising and baseline correction by discrete wavelet transform for microchip capillary electrophoresis., *Electrophoresis* **24**(18): 3260–3265.

Liu, Q., Krishnapuram, B., Pratapa, P., Liao, X., Hartemink, A. and Carin, L. (2003). Identification of differentially expressed proteins using maldi-tof mass spectra, *Conference Record of the Thirty-Seventh Asilomar Conference on Signals, Systems and Computers*, Vol. 2, pp. 1323– 1327.

Liu, W., Guan, M., Wu, D., Zhang, Y., Wu, Z., Xu, M. and Lu, Y. (2005). Using tree analysis pattern and seldi-tof-ms to discriminate transitional cell carcinoma of the bladder cancer from noncancer patients., *Eur Urol* **47**(4): 456–462.

Lo, A. Y. (2005). Weighted chinese restaurant processes, *Cosmos* **1**(1): 107–111. World Scientific Publishing Company.

Louis, A., Maass, P. and Rieder, A. (1998). *Wavelets: Theorie und Anwendungen*, 2nd edn, B.G. Teubner, Stuttgart.

M. J. Litzkow, M. L. and Mutka, M. W. (1988). Condor-a hunter of idle workstations, pp. 104–111.

Ma, B., Zhang, K., Hendrie, C., Liang, C., Li, M., Doherty-Kirby, A. and Lajoie, G. (2003). Peaks: powerful software for peptide de novo sequencing by tandem mass spectrometry, *Rapid Communication in Mass Spectrometry* **17**: 2337–2342.

Mäkinen, V. (2007). Peak alignment using restricted edit distances, *Biomolecular Engineering* **24**(3): 337–342.

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

Martin, B. (1995). *Instance-based learning : Nearest neighbor with generalization*, Master's thesis, University of Waikato, Hamilton, New Zealand.

Maschke, K. J. (2005). Navigating an ethical patchwork–human gene banks., *Nat Biotechnol* **23**(5): 539–545.

Mathis, M., Semke, J. and Mahdavi, J. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm, *Computer Communications Review* **27**(3).

Matsuoka, S., Ballif, B. A., Smogorzewska, A., McDonald, E. R., Hurov, K. E., Luo, J., Bakalarski, C. E., Zhao, Z., Solimini, N., Lerenthal, Y., Shiloh, Y., Gygi, S. P. and Elledge, S. J. (2007). Atm and atr substrate analysis reveals extensive protein networks responsive to dna damage., *Science* **316**(5828): 1160–1166.

Mazet, V., Brie, D. and Idier, J. (2004). Baseline spectrum estimation using half-quadratic minimization, *Proceedings of the European Signal Processing Conference*, Vienna, Autriche.

McCloskey, D. (1995). The insignificance of statistical significance, *Scientific American* **72**: 32–33.

McDonald, K. (2006). How big blue created the blue gene and is now delving into the blue brain, Australian Life Scientist, Online Journal `www.biotechnews.com.au/index.php/id;152112833;fp;4194304;fpid;1`, accessed October 11, 2007.

McDonough, R. N. and Whale, A. D. (1995). *Detection of Signals in Noise*, 2nd edn, Academic Press, San Diego.

McLachlan, S. M. and Rapoport, B. (2004). Why measure thyroglobulin autoantibodies rather than thyroid peroxidase autoantibodies?, *Thyroid* **14**(7): 510–520.

McLean, J. and Ernest, J. (1998). The role of statistical significance testing in educational research, *Research in the Schools* **5**(2): 15–22.

Mehlhorn, K. and Naher, S. (1999). *LEDA: A Platform for Combinatorial and Geometric Computing*, Cambridge University Press.

Meyer, F. (1979). *Cytologie quantitative et morphologie mathmatique*, . thse de docteur-ingnieur, Ecole des Mines de Paris.

Mitchell, J. (2003). Measurement: A beginner's guide, *Journal of Applied Measurement* **4**(4): 298–308.

Mueller, J., von Eggeling, F., Driesch, D., Schubert, J., Melle, C. and Junker, K. (2005). Proteinchip technology reveals distinctive protein expression profiles in the urine of bladder cancer patients., *Eur Urol* **47**(6): 885–93; discussion 893–4.

Mueller, L. N., Rinner, O., Schmidt, A., Letarte, S., Bodenmiller, B., Brusniak, M.-Y., Vitek, O., Aebersold, R. and Mller, M. (2007). Superhirn - a novel tool for high resolution lc-ms-based peptide/protein profiling., *Proteomics* **7**(19): 3470–3480.

Munkres, J. (1957). Algorithms for the assignment and transportation problems, *Journal of the Society of Industrial and Applied Mathematics* **5**(1): 32–38.

Mrtz, E., O'Connor, P. B., Roepstorff, P., Kelleher, N. L., Wood, T. D., McLafferty, F. W. and Mann, M. (1996). Sequence tag identification of intact proteins by matching tanden mass spectral data against sequence data bases., *Proc Natl Acad Sci U S A* **93**(16): 8264–8267.

Naaby-Hansen, S., Warnasuriya, G. D., Hastie, C., Gallney, P. and Cramer, R. (2005). Proteomic approaches in the analysis of hypertension., *Methods Mol Med* **108**: 275–296.

Nason, G. P. and Silverman, B. W. (1995). The stationary wavelet transform and some statistical applications, *Lecture Notes in Statistics*, Vol. 103, pp. 281–300.

Neyman, J. and Pearson, E. S. (1928). On the use and interpretation of certain test criteria for purposes of statistical inference, *Biometrika* **20A**: 175–240,263–94.

Nightingale, P. and Martin, P. (2004). The myth of the biotech revolution., *Trends Biotechnol* **22**(11): 564–569.

Norris, J. L., Cornett, D. S., Mobley, J. A., Schwartz, S. A., Roder, H. and Caprioli, R. M. (2005). Preparing maldi mass spectra for statistical analysis: A practical approach, *Proceedings of the 53rd ASMS Conference on Mass Spectrometry and Allied Topics*, San Antonio, TX.

Ojanen, J., Miettinen, T., Heikkonen, J. and Rissanen, J. (2004). Robust denoising of electrophoresis and mass spectrometry signals with minimum description length principle., *FEBS Lett* **570**(1-3): 107–113.

O'Leary, P. C., Feddema, P. H., Michelangeli, V. P., Leedman, P. J., Chew, G. T., Knuiman, M., Kaye, J. and Walsh, J. P. (2006). Investigations of thyroid hormones and antibodies based on a community health survey: the busselton thyroid study, *Clinical Endocrinology* **64**: 97–104(8).

Omenn, G. S. (2006). Strategies for plasma proteomic profiling of cancers., *Proteomics* **6**(20): 5662–5673.

Ong, S.-E. and Mann, M. (2005). Mass spectrometry-based proteomics turns quantitative., *Nat Chem Biol* **1**(5): 252–262.

OpenMS (2008). Openms homepage, `http://open-ms.sourceforge.net/OpenMS.php`, accessed October 11, 2007.

O'Reilly, T., Peek, J. and Loukides, M. (1997). *UNIX Power Tools, 2nd edition*, O'Reilly & Assoc. Inc., Sebastopol, California.

Paalanen, P., Kamarainen, J.-K., Ilonen, J. and Kälviäinen, H. (2005). Representation and discrimination based on gaussian mixture model probability densities - practices and algorithms, *Technical Report 95*, Lappeenranta University of Technology, Department of Information Technology.

Palsson, G. and Rabinow, P. (1999). Iceland: the case of a national human genome project., *Anthropol Today* **15**(5): 14–18.

Pamuk, O. N. and Cakir, N. (2007). The frequency of thyroid antibodies in fibromyalgia patients and their relationship with symptoms., *Clin Rheumatol* **26**(1): 55–59.

Pandey, A. and Mann, M. (2000). Proteomics to study genes and genomes., *Nature* **405**(6788): 837–846.

Papadimitriou, C. H. and Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Pappin, D., Hojrup, P. and Bleasby, A. (1993). Rapid identification of proteins by peptide-mass fingerprinting., *Curr Biol* **3**(6): 327–32.

Pavlidis, T. (2003). 36 years on the pattern recognition front lecture given at icpr'2000 in barcelona, spain on the occasion of receiving the k.s. fu prize, *Pattern Recogn. Lett.* **24**(1-3): 1–7.

Perkins, D., Pappin, J., Creasy, D. and Cottrell, J. (1999). Probability-based protein identification by searching database using mass spectrometry data, *Electrophoresis* **20**: 3551–3567.

Petricoin, E. F., Ardekani, A. M., Hitt, B. A., Levine, P. J., Fusaro, V. A., Steinberg, S. M., Mills, G. B., Simone, C., Fishman, D. A., Kohn, E. C. and Liotta, L. A. (2002). Use of proteomic patterns in serum to identify ovarian cancer., *Lancet* **359**(9306): 572–577.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization, pp. 185–208.

Potter, R. W. (2000). *The Art of Measurement: Theory and Practice*, Prentice Hall, Upper Saddle River, NJ.

Qian, W.-J., Jacobs, J. M., Liu, T., Camp, D. G. and Smith, R. D. (2006). Advances and challenges in liquid chromatography-mass spectrometry-based proteomics profiling for clinical applications., *Mol Cell Proteomics* **5**(10): 1727–1744.

Qu, Y., Adam, B.-L., Yasui, Y., Ward, M. D., Cazares, L. H., Schellhammer, P. F., Feng, Z., Semmes, O. J. and Wright, G. L. (2002). Boosted decision tree analysis of surface-enhanced laser desorption/ionization mass spectral serum profiles discriminates prostate cancer from noncancer patients., *Clin Chem* **48**(10): 1835–1843.

Radetti, G., Gottardi, E., Bona, G., Corrias, A., Salardi, S., Loche, S., for Thyroid Diseases of the Italian Society for Pediatric Endocrinology, S. G. and (SIEDP/ISPED), D. (2006). The natural history of euthyroid hashimoto's thyroiditis in children., *J Pediatr* **149**(6): 827–832.

Ranish, J. A., Hahn, S., Lu, Y., Yi, E. C., jun Li, X., Eng, J. and Aebersold, R. (2004). Identification of tfb5, a new component of general transcription and dna repair factor iih., *Nat Genet* **36**(7): 707–713.

Reich, M., Liefeld, T., Gould, J., Lerner, J., Tamayo, P. and Mesirov, J. P. (2006). Genepattern 2.0., *Nat Genet* **38**(5): 500–501.

Rivest, R. L., Shamir, A. and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems, *Commun. ACM* **21**(2): 120–126.

Rosenblatt, K. P., Bryant-Greenwood, P., Killian, J. K., Mehta, A., Geho, D., Espina, V., Petricoin, E. F. and Liotta, L. A. (2004). Serum proteomics in cancer diagnosis and management., *Annu Rev Med* **55**: 97–112.

Rosenthal, R. and Rosnow, R. (1991). *Essentials of behavioral research: Methods and data analysis*, 2nd edn, Mcgraw-Hill, New York.

Rosnow, R. and Rosenthal, R. (1996). Computing contrasts, effect sizes, and counternulls on other people's published data: General procedures for research consumers, *Pyschological Methods* **1**: 331–340.

Rubner, Y., Tomasi, C. and Guibas, L. (2000). The earth mover's distance as a metric for image retrieval, *International Journal of Computer Vision* **40**(2): 99–121.

Rudy, E. B. and Kerr, M. (1991). Unraveling the mystique of power analysis., *Heart Lung* **20**(5 Pt 1): 517–522.

S. Kullback, R. L. (1951). On information and sufficiency, *Annals of Mathematical Statistics* **22**: 79–86.

Safarik, I. and Safarikova, M. (2004). Magnetic techniques for the isolation and purification of proteins and peptides., *Biomagn Res Technol* **2**(1): 7.

Sauve, A. C. and Speed, T. P. (2004). Normalization, baseline correction and alignment of high-throughput mass spectrometry data, *Procedings Gensips 2004*.

Scheather, S. J. (2004). Density estimation, *Statistical Science* **19**(4): 588–597.

Schmid, H., Semjonow, A., Stieber, P. and van Poppel, H. (1999). Tumour markers in germ cell cancer: Egtm recommendations. european group on tumour markers., *Anticancer Res* **19**(4A): 2795–2798.

Serra, J. (1982). *Image Analysis and Mathematical Morphology*, Academic Press, London.

Shao, J. and Tu, D. (1995). *The Jackknife and Bootstrap*, Springer.

Shapiro, S. S. and Wilk, M. B. (1965). An analysis of variance test for normality (complete samples), *Biometrika* **52**(3): 591–611.

Shaver, J. P. (1985). Chance and nonsense: A conversation about interpreting tests of statistical significance, *Phi Delta Kappan* **67**(2): 138–141.

Shepard, R. (1962). The analysis of proximities: Multidimensional scaling with an unknown distance function., *I. Psychometrika* .

Shin, H., Mutlu, M., Koomen, J. M. and Markey, M. K. (2007). Parametric power spectral density analysis of noise from instrumentation in maldi tof mass spectrometry, *Cancer Informatics* **3**: 317–328.

Sinnott, R. W. (1984). Virtues of the haversine, *Sky and Telescope* **68**(2): 159.

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

Smyth, P. P. A., Wijeyaratne, C. N., Kaluarachi, W. N., Smith, D. F., Premawardhana, L. D. K. E., Parkes, A. B., Jayasinghe, A., de Silva, D. G. H. and Lazarus, J. H. (2005). Sequential studies on thyroid antibodies during pregnancy., *Thyroid* **15**(5): 474–477.

Statheropoulos, M., Pappaa, A., Karamertzanisa, P. and Meuzelaarb, H. L. C. (1999). Noise reduction of fast, repetitive gc/ms measurements using principal component analysis (pca), *Analytica Chimica Acta* **401**(1-2): 35–43.

Stein, C. (1981). Estimation of the mean of a multivariate normal distribution., *Annals of Statistics* **9**(6): 1135–1151.

Sterling, T., Savarese, D., Becker, D. J., Dorband, J. E., Ranawake, U. A. and Packer, C. V. (1995). BEOWULF: A parallel workstation for scientific computation, *Proceedings of the 24th International Conference on Parallel Processing*, Oconomowoc, WI, pp. I:11–14.

Stevens, R. D., Robinson, A. J. and Goble, C. A. (2003). mygrid: personalised bioinformatics on the information grid., *Bioinformatics* **19 Suppl 1**.

Stone, M. (1974). Cross-validatory choice and assessment of statistical predictions (with discussion), *Journal of the Royal Statisticul Society* **36**: 111–147.

SUNMicrosystems (2006). Documentation of the java platform standard edition 6, `http://http://java.sun.com/javase/6/docs/api/`, accessed October 11, 2007.

Suriano, R., Lin, Y., Ashok, B. T., Schaefer, S. D., Schantz, S. P., Geliebter, J. and Tiwari, R. K. (2006). Pilot study using seldi-tof-ms based proteomic profile for the identification of diagnostic biomarkers of thyroid proliferative diseases., *J Proteome Res* **5**(4): 856–861.

Süssmuth, R. D. and Jung, G. (1999). Impact of mass spectrometry on combinatorial chemistry., *J Chromatogr B Biomed Sci Appl* **725**(1): 49–65.

Tammen, H., Schulte, I., Hess, R., Menzel, C., Kellmann, M., Mohring, T. and Schulz-Knappe, P. (2005). Peptidomic analysis of human blood specimens: Comparison between plasma specimens and serum by differential peptide display, *Proteomics* **5**(13): 3414–3422.

Tang, N., Tornatore, P. and Weinberger, S. (2004). Current developments in seldi affinity technology, *Mass spectrometry reviews* **23**(1): 34–33.

Taswell, C. (2000). The what, how, and why of wavelet shrinkage denoising, *Computing in Science and Engineering* **2**(3): 12–19.

Terry, S. F., Terry, P. F., Rauen, K. A., Uitto, J. and Bercovitch, L. G. (2007). Advocacy groups as research organizations: the pxe international example., *Nat Rev Genet* **8**(2): 157–164.

Thadikkaran, L., Siegenthaler, M. A., Crettaz, D., Queloz, P.-A., Schneider, P. and Tissot, J.-D. (2005). Recent advances in blood-related proteomics., *Proteomics* **5**(12): 3019–3034.

Thompson, B. (1994). The concept of statistical significance testing, *Practical Assessment, Research & Evaluation* **4**(5).

Thompson, B. (1995). Exploring the replicability of a study's results: Bootstrap statistics for the multivariate case, *Educational and Psychological Measurement* **55**: 84–94.

Thulasidasan, S., chun Feng, W. and Gardner, M. K. (2003). Optimizing gridftp through dynamic right-sizing, *hpdc* **00**: 14.

Tibshirani, R., Hastie, T., Narasimhan, B., Soltys, S., Shi, G., Koong, A. and Le, Q.-T. (2004). Sample classification from protein mass spectrometry, by 'peak probability contrasts'., *Bioinformatics* **20**(17): 3034–3044.

Tolson, J., Bogumil, R., Brunst, E., Beck, H., Elsner, R., Humeny, A., Kratzin, H., Deeg, M., Kuczyk, M., Mueller, G. A., Mueller, C. A. and Flad, T. (2004). Serum protein profiling by seldi mass spectrometry: detection of multiple variants of serum amyloid alpha in renal cancer patients., *Lab Invest* **84**(7): 845–856.

Torres-Cabala, C., Bibbo, M., Panizo-Santos, A., Barazi, H., Krutzsch, H., Roberts, D. D. and Merino, M. J. (2006). Proteomic identification of new biomarkers and application in thyroid cytology., *Acta Cytol* **50**(5): 518–528.

Tukey, J. (1958). Bias and confidence in not-quite large samples, *Annals of Mathematical Statistics* **29**: 614.

Tyler, R. (1931). What is statistical significance?, **10**(5): 115–142.

Veltkamp, R. C. and Hagedoorn, M. (2000). Shape similarity measures, properties and constructions, *VISUAL '00: Proceedings of the 4th International Conference on Advances in Visual Information Systems*, Springer-Verlag, London, UK, pp. 467–476.

Verbeek, J. J., Vlassis, N. and Kröse, B. (2003). Efficient greedy learning of gaussian mixture models., *Neural Comput* **15**(2): 469–485.

Villanueva, J., Martorella, A. J., Lawlor, K., Philip, J., Fleisher, M., Robbins, R. J. and Tempst, P. (2006). Serum peptidome patterns that distinguish metastatic thyroid carcinoma from cancer-free controls are unbiased by gender and age., *Mol Cell Proteomics* **5**(10): 1840–1852.

Villanueva, J., Shaffer, D. R., Philip, J., Chaparro, C. A., Erdjument-Bromage, H., Olshen, A. B., Fleisher, M., Lilja, H., Brogi, E., Boyd, J., Sanchez-Carbayo, M., Holland, E. C., Cordon-Cardo, C., Scher, H. I. and Tempst, P. (2006). Differential exoprotease activities confer tumor-specific serum peptidome patterns., *J Clin Invest* **116**(1): 271–284.

Vlahou, A., Schellhammer, P. F., Mendrinos, S., Patel, K., Kondylis, F. I., Gong, L., Nasim, S. and Jr, G. L. W. (2001). Development of a novel proteomic approach for the detection of transitional cell carcinoma of the bladder in urine., *Am J Pathol* **158**(4): 1491–1502.

Vyssotsky, V. A., Corbat, F. J. and Graham, R. M. (1965). Introduction and overview of the multics system, *Proceedings of Fall Joint Computer Conference*, Vol. 203 of *AFIPS Conf. Proc.*

Wagner, M., Naik, D. and Pothen, A. (2003). Protocols for disease classification from mass spectrometry data., *Proteomics* **3**(9): 1692–1698.

Wang, G., Wu, W., Pisitkun, T., Hoffert, J., Knepper, M. and Shen, R.-F. (2006). Automated quantification tool for high-throughput proteomics using stable isotope labeling and LC-MS, *Analytical Chemistry* **78**(16): 5752–5761.

Wang, M. Z., Howard, B., Campa, M. J., Patz, E. F. and Fitzgerald, M. C. (2003). Analysis of human serum proteins by liquid phase isoelectric focusing and matrix-assisted laser desorption/ionization-mass spectrometry., *Proteomics* **3**(9): 1661–1666.

Wang, W., Zhou, H., Lin, H., Roy, S., Shaler, T., Hill, L., Norton, S., Kumar, P., Anderle, M. and Becker, C. (2003). Quantification of proteins and metabolites by mass spectrometry without isotopic labeling or spiked standards, *Analytical Chemistry* **75**(18): 4818–4826.

Washburn, M. P., Wolters, D. and Yates, J. R. (2001). Large-scale analysis of the yeast proteome by multidimensional protein identification technology., *Nat Biotechnol* **19**(3): 242–247.

Wasinger, V. C., Cordwell, S. J., Cerpa-Poljak, A., Yan, J. X., Gooley, A. A., Wilkins, M. R., Duncan, M. W., Harris, R., Williams, K. L. and Humphery-Smith, I. (1995). Progress with gene-product mapping of the mollicutes: Mycoplasma genitalium., *Electrophoresis* **16**(7): 1090–1094.

Wasserman, L. (2006). *All of Nonparametric Statistics (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Wells, J. M. and McLuckey, S. A. (2005). Collision-induced dissociation (cid) of peptides and proteins., *Methods Enzymol* **402**: 148–185.

WfMC (2007). Workflow management coalition reference model, `http://www.wfmc.org`, accessed October 11, 2007.

Whisstock, J. C. and Lesk, A. M. (2003). Prediction of protein function from protein sequence and structure., *Q Rev Biophys* **36**(3): 307–340.

Wilkins, M. R., Pasquali, C., Appel, R. D., Ou, K., Golaz, O., Sanchez, J. C., Yan, J. X., Gooley, A. A., Hughes, G., Humphery-Smith, I., Williams, K. L. and Hochstrasser, D. F. (1996). From proteins to proteomes: large scale protein identification by two-dimensional electrophoresis and amino acid analysis., *Biotechnology (N Y)* **14**(1): 61–65.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*, 2nd edn, Morgan Kaufmann, San Francisco.

Wittmann and Heinzle (1999). Mass spectrometry for metabolic flux analysis., *Biotechnol Bioeng* **62**(6): 739–750.

Wolfson, H. J. and Rigoutsos, I. (1997). Geometric hashing: An overview, *IEEE Computational Science & Engineering* **4**(4): 10–21.

xia Guan, H., yang Li, C., shu Li, Y., ling Fan, C., Teng, Y., hong Ouyang, Y., Cong, Q. and ping Teng, W. (2006). [thyroid function and thyroid autoimmunity at the late pregnancy: data from 664 pregnant women], *Zhonghua Fu Chan Ke Za Zhi* **41**(8): 529–532.

Yang, J. and Honavar, V. G. (1998). Feature subset selection using a genetic algorithm, *IEEE Intelligent Systems* **13**(2): 44–49.

Zaletel, K., Krhin, B., Gaberscek, S. and Hojker, S. (2006). Thyroid autoantibody production is influenced by exon 1 and promoter ctla-4 polymorphisms in patients with hashimoto's thyroiditis., *Int J Immunogenet* **33**(2): 87–91.

Zarbock, R., Hendig, D., Szliska, C., Kleesiek, K. and Gtting, C. (2007). Pseudoxanthoma elasticum: genetic variations in antioxidant genes are risk factors for early disease onset., *Clin Chem* **53**(10): 1734–1740.

Zeng, M., Li, J. and Peng, Z. (2006). The design of top-hat morphological filter and application to infrared target detection, *Infrared Physics and Technology* **48**: 67–76.

Zhang, J., Gao, W., Cai, J., He, S., Zeng, R. and Chen, R. (2005). Predicting molecular formulas of fragment ions with isotope patterns in tandem mass spectra, *IEEE/ACM Transactions on Computational Biology and Bioinformatics* **02**(3): 217–230.

Zhang, X., Wei, D., Yap, Y., Li, L., Guo, S. and Chen, F. (2007). Mass spectrometry-based omics technologies in cancer diagnostics, *Mass Spectrometry Reviews* **26**(3): 403–431.

Zhang, Z., Bast, R. C., Yu, Y., Li, J., Sokoll, L. J., Rai, A. J., Rosenzweig, J. M., Cameron, B., Wang, Y. Y., Meng, X.-Y., Berchuck, A., Haaften-Day, C. V., Hacker, N. F., de Bruijn, H. W. A., van der Zee, A. G. J., Jacobs, I. J., Fung, E. T. and Chan, D. W. (2004). Three biomarkers identified from serum proteomic analysis for the detection of early stage ovarian cancer., *Cancer Res* **64**(16): 5882–5890.

Zphel, K., Saller, B., Wunderlich, G., Grning, T., Koch, R., Wilde, J., Mann, K. and Franke, W.-G. (2003). Autoantibodies to thyroperoxidase (tpoab) in a large population of euthyroid subjects: implications for the definition of tpoab reference intervals., *Clin Lab* **49**(11-12): 591–600.