SPIDER: Reconstructive Protein Homology Search with *De Novo* Sequencing Tags

by

Denis Yuen

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2011

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In the field of proteomic mass spectrometry, proteins can be sequenced by two independent yet complementary algorithms: *de novo* sequencing which uses no prior knowledge and database search which relies upon existing protein databases. In the case where an organism's protein database is not available, the software Spider was developed in order to search sequence tags produced by *de novo* sequencing against a database from a related organism while accounting for both errors in the sequence tags and mutations.

This thesis further develops Spider by using the concept of reconstruction in order to predict the real sequence by considering both the sequence tags and their matched homologous peptides. The significant value of these reconstructed sequences is demonstrated. Additionally, the runtime is greatly reduced and separated into independent caching and matching steps.

This new approach allows for the development of an efficient algorithm for search. In addition, the algorithm's output can be used for new applications. This is illustrated by a contribution to a complete protein sequencing application.

Acknowledgements

First, I would like to thank my parents and Pei Wang. I would like to acknowledge my supervisor Bin Ma, as well as my co-workers for their support.

Second, I would also like to thank Xiaowen Liu and Yonghua Han for their major contributions to the initial development of Spider and Champs software respectively.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

It has been said that the "goal of proteomics is to identify and quantify all the proteins present in a cell at a specific moment."[1] This concise description encapsulates two of the major challenges in the related tasks of protein identification and quantification by tandem mass spectrometry (MS/MS). While MS/MS is widely utilized in biology labs, the basic problems of identifying unknown proteins and how much of those unknown proteins are present in a biological sample are still challenges. Existing software tools and techniques can have very different approaches and there is strong competition for the best performance.

This thesis will address the problem of identifying (and sequencing) unknown proteins which are not contained within a protein database but are homologous[1] to proteins that are.

1.1 Thesis Statement

In protein identification, there are two main approaches to identifying peptides in MS/MS data. The first approach is to match the data directly to a protein sequence database. However, this approach requires the existence of relevant protein sequence databases. In this context, a relevant database is one that is derived from the same species that the biological sample came from. In the case that this is not possible, one approach is to use *de novo* sequencing in order to create the peptide sequence without the aid of a database. However, this results in peptide sequences that have no relationships to any proteins. Additionally, very often the *de novo* sequencing results are not completely accurate.

The next step after this process, if a protein database for a related species is available, is the process of homology search. Spider [3] was a tool designed for this task, taking into account *de novo* sequencing errors and matching the *de novo* sequence tags with potentially homologous peptides. This allows us to identify (and provide coverage of) proteins that are homologous to proteins in our sample.

This thesis will explore and evaluate the creation of a new version of Spider, referred to as Spider II, designed to reconstruct the "true" peptide sequence after taking into account sequencing errors and

[1] It has been noted that there is a distinction between sequence similarity and homology. Two sequences can be homologous but not similar. Conversely, they can be similar and not homologous [2]. Briefly, two sequences are homologous if they share a common ancestor and evidence of this is significant sequence similarity. However, similarity and homology are correlated. As there is no effective way to determine homology, bioinformatics researchers usually use similarities in lieu of homologies.

mutations. In addition, Spider II will also allow searching for variable PTMs (post-translational modifications). Finally, this thesis will also evaluate some applications of this process that can be used to sequence whole unknown proteins.

The motivation for this thesis is analogous to the traditional reasons that researchers in bioinformatics or related fields use homology search, to characterize unknown proteins, to discover shared characteristics, and to put proteins in some kind of evolutionary context. Ultimately, it is our hope that Spider will enable and lead to tools that will allow researchers specifically in the field of mass spectrometry to do the same.

1.2 Outline

Chapter 2 will cover the basics of molecular biology and tandem mass spectrometry in order to provide the background for what we are trying to accomplish. Chapter 3 will define the homologous protein search, reconstruction, and protein sequencing problems. Existing approaches will also be reviewed. Chapter 4 will describe the methodology and the results respectively for the new reconstructive Spider search. Chapter 5 will describe the methodology and results for Champs, an extension of Spider reconstruction. Finally, Chapter 6 will give conclusions and outline suggestions for future work.

Chapter 2
Background

In this chapter, we will provide basic information on molecular biology, leading up to proteomics and mass spectrometry, allowing us to discuss tools like database search and *de novo* sequencing.

2.1 Central Dogma of Molecular Biology

The first statement of the central dogma of molecular biology was stated as follows by Francis Crick in 1958:

> The central dogma of molecular biology deals with the detailed residue-by-residue transfer of sequential information. It states that information cannot be transferred back from protein to either protein or nucleic acid.[4]

This provides a good point of introduction into the molecular biology required to understand this thesis. In short, there are three types of biological sequences, DNA (Deoxyribonucleic Acid), RNA (Ribonucleic Acid), and proteins (amino acid sequences). As depicted in Figure 1, we can see that information flows forward from DNA, to RNA, and to proteins with the less common exception of viral reverse transcription.

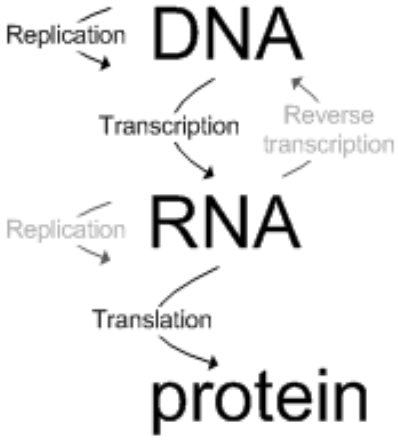

Figure 1: Central Dogma of Molecular Biology

DNA is composed of four nucleic acids (A, T, C, and G)[2] which are transcribed into RNA (A, U, C, and G)[3] which is subsequently translated into a peptide sequence. In general, peptides are formed from 20 amino acids. Peptide sequences then fold into a three dimensional protein structure. This process is constantly occurring in the cell. Thus, DNA and RNA respectively store information and act as a template for the construction of proteins which perform various functions in living organisms. By understanding which proteins (and how much of them)[4] are functioning in a cell or in a human body at any given time, scientists hope to determine how proteins are responsible for particular conditions such as immune response to disease, cell growth, and so forth.

One important concept to understand is that given a specific nucleotide sequence with a known reading frame, DNA is translated in a predictable way[5] to specific amino acids. Each amino acid has a unique mass with the exception of Leucine (L) and Isoleucine (I) which share the same mass. This property will become important later when we explore mass spectrometry in the field of proteomics.

Another complicating factor is that amino acids can be modified after translation in a process called post-translational modification (PTM). Common examples include phosphorylation, carboxymethylation, and oxidation. These PTMs often control the behaviour of proteins by affecting different biological tasks within the cell, such as triggering the regulation of transcription, protecting proteins from proteolytic enzymes, and so forth. From a computational perspective, this essentially creates additional amino acids which may or may not have unique masses which can complicate the process of analysis. The last complicating factor is that in majority of cases, protein databases do not include information on PTMs which makes it more difficult to analyze peptides when using a database to aid analysis.

[2] Adenine, thymine, cytosine, and guanine
[3] Adenine, uracil, cytosine, and guanine
[4] For those reading ahead, those two questions give rise to the problems of protein identification and protein quantification
[5] Of course it can be difficult to find the genes in a first place and in eukaryote organisms, to predict splicing

2.2 Proteomics and Mass Spectrometry

Traditionally, in order to sequence proteins, a chemical technique known as Edman degradation had to be used [5]. Unfortunately, this method was incompatible with certain post-translational modifications and was also quite slow since individual amino acids have to be labelled and cleaved one-by-one.

As a result, tandem mass spectrometry has become a faster and more popular alternative. In this approach, we take a protein (or usually a mixture of proteins) in a sample obtained from the lab and digest it with an enzyme, breaking the proteins into many smaller peptides with endpoints that depend on the type of enzyme, its efficiency, and its specificity. These peptides are ionized and their movement measured in order to obtain a MS scan which reveals the mass of these smaller peptides[6]. These peptides are then fragmented in a number of different ways to yield a MS/MS scan which reveals the mass of different pieces of the individual peptides. MS/MS scans result in a mass spectrum indexed by mass and measuring intensity. The individual points in these scans are commonly referred to as "peaks." This means that a mass spectrometer will contain at least three components, an ionizer to create an ion source, a mass analyzer to differentiate between ions of different mass-to-charge ratios, and a detector to detect the ions. These ions are called precursor ions in the MS scans and fragment ions in the MS/MS scans.

[6] Technically, these instruments only measure the mass to charge ratios (m/z) of the ions associated with particular peptides. Luckily, we can decipher the charge state based on the distribution of the ions.

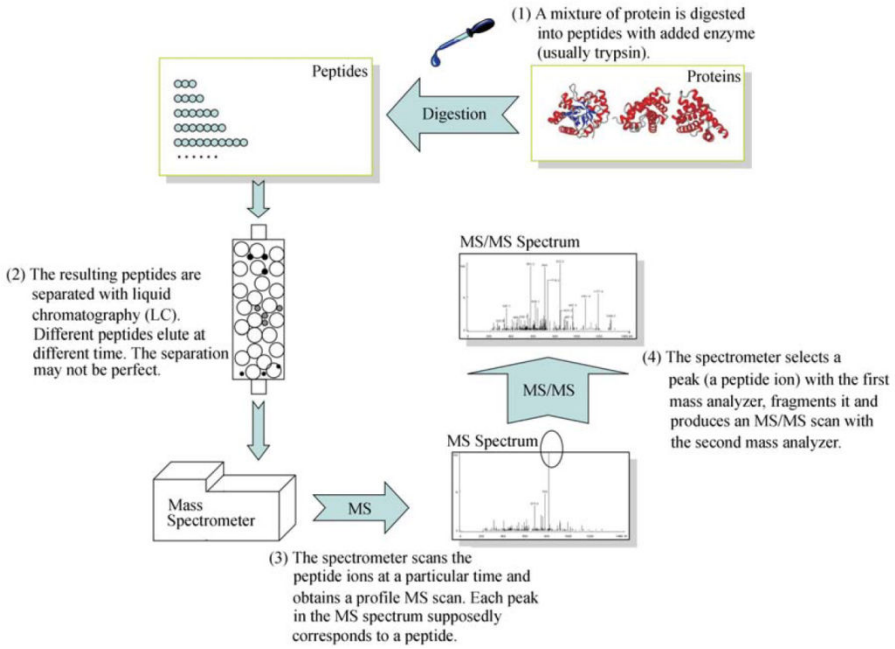

Figure 2: High-level workflow depicting mass spectrometry experiment [6]

A summary of the whole process is depicted by Figure 2 although there are many subtleties and variations that have been introduced. For example, we can use multiple enzymes to obtain multiple overlapping peptides from each of the different enzymes. Trypsin cuts after a K (lysine) or R (arginine), but not before P (proline) while GluC cuts after D (aspartic acid) or E (glutamic acid), but not before P[7]. There are also methods to provide separation between the sample preparation and ionization stages. For example, liquid chromatography or high performance liquid chromatography (LC or HPLC) can be used to separate the sample mixture by the hydrophobicity of peptides within it. Another approach would be to use SDS-PAGE (Sodium Dodecyl Sulfate Polyacrylamide Gel Electrophoresis) which separates peptides by their electrophoretic mobility which is proportional to their ability to travel through the gel. This mobility is related to a particular peptide's length and molecular weight.

It should be noted that this is only a brief summary of the varieties of equipment that are used in the mass spectrometry process. There are a variety of ion sources, the most popular in this field being electrospray ionization (ESI) and matrix assisted laser desorption/ionization (MALDI), and a large variety of mass analyzers including time-of-flight (TOF), linear ion trap, Fourier transform mass spectrometry (FTMS), Orbitrap, and quadrupole mass analyzers.

[7] Past this point, please refer to Table 1 in order to map amino acid codes with their full names

There are a great number of variants in this overall process since detectors can be combined in various ways to produce both a MS and MS/MS scan as shown in Figure 3. For example, a QTOF uses a quadrupole to do the MS and a TOF to do the MS/MS. The LTQ FT and LTQ Orbitrap use a FT and Orbitrap for the MS respectively and a LTQ (Linear Ion Trap) for the MS/MS scan. Peptides can also be fragmented in a number of different ways including collision-induced dissociation (CID), electron-capture dissociation (ECD), and electron-transfer dissociation (ETD). The large number of variations creates a large spectrum of possible results with different mass accuracies and cost-effectiveness. Different fragmentation types also produce different amounts of ions and different kinds of ions.

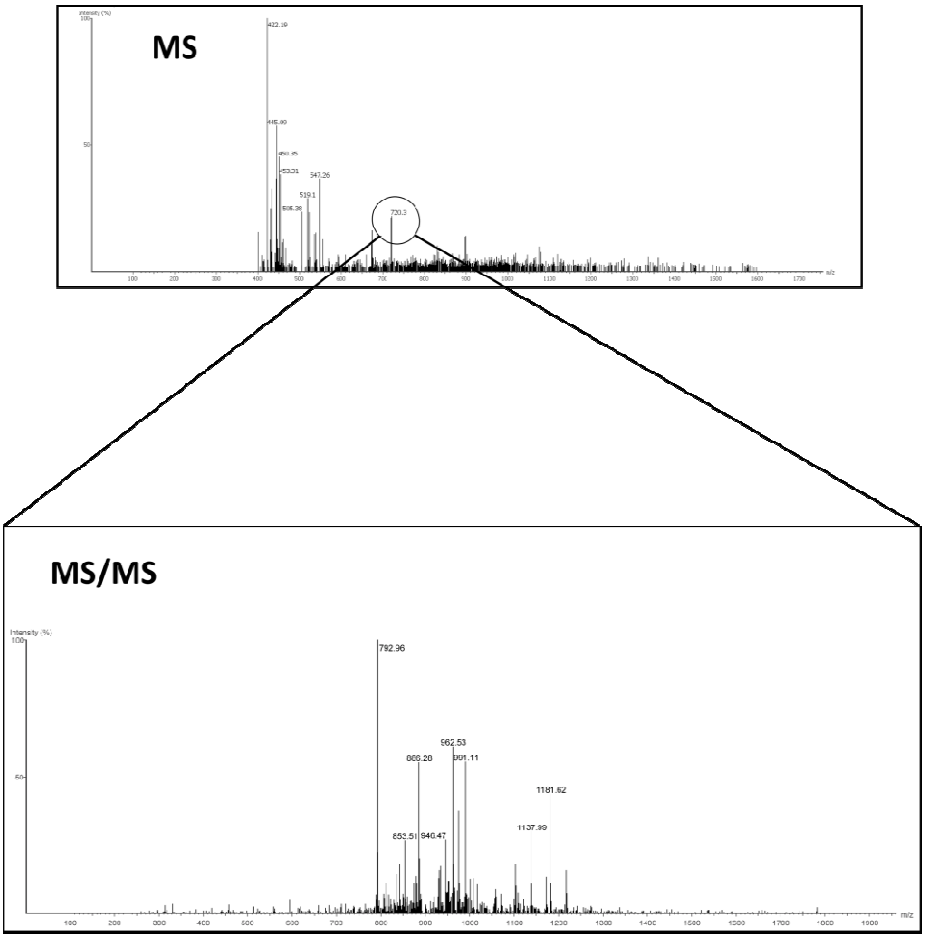

Figure 3: Relationship between MS and MS/MS scans

Peaks (as shown in Figure 3 as a series of bars) come from what are referred to as b-ion fragments or y-ion fragments. By convention, these fragments are produced when the charge is retained either

on the amino-terminal fragment or when the charge is retained by the carboxyl-terminal fragment respectively (It should be noted that an alternate popular name for these fragments or for the amino acids at these two terminals are "N-terminal" and "C-terminal" named for the NH_3^+ and COO^- groups at either end). Since we know we have some understanding of how the peptides fragment and create these peaks, we can deduce the sequence of the original peptide by two methods which will be detailed in the next section.

Unfortunately, while sounding easy, this quick summary overlooks issues that may complicate the analysis such as expected or unexpected modifications, missed or unexpected enzyme cleavages, unusual fragmentation, sample contamination, and instrument noise [7]. Additionally, we can expect that even in the case where these kinds of problems are absent, individual peaks can be split into a group of isotopic peaks (due to different amounts of various isotopes of C being used) or different types of fragment ions (a, c, x, or z fragments). Another difficulty is the possibility of internal cleavage ions when the ions that are created are caused by multiple cleavages and are not anchored at one of the two terminal ends. Figure 4 demonstrates how these ions are created as fragment ions from a full peptide. Y and b ions are labelled from the carboxyl-terminal end and the amino-terminal end respectively. They are numbered by the number of amino-acid R groups that are contained within the fragment ion.

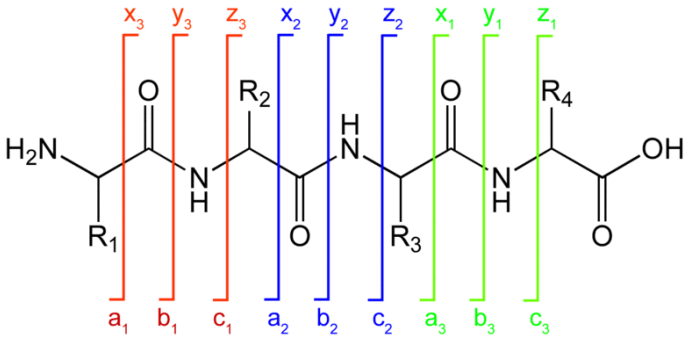

Figure 4: Depicting general chemical structure [8]

Taking all these difficulties into account, much of the data that is generated by the mass spectrometer is not useful for identification and much of what is identified is of dubious confidence. From the perspective of mass spectrometer data, this problem is especially prominent when less reliable and less expensive instruments with lower mass tolerances and/or more noise are used. From the perspective of proteins, the result is that in many cases only a low percentage of the proteins in a sample are mapped by identified peptides. This leaves many proteins uncharacterized or inadequately

covered [9]. This gap explains the large number of competing software approaches to the problem which can compete based on improved sensitivity (while balancing specificity) and performance (CPU-time or memory usage).

2.3 *De Novo* Sequencing

The phrase *de novo* is Latin for "from the beginning" or "anew." In the context of bioinformatics, it is used to describe computational approaches that do not require comparison with related data. *De novo* peptide sequencing indicates that we do not need the aid of a protein database. Another example would be *de novo* protein structure prediction where tertiary protein structure is modeled without the aid of a template structure.

In any case, the first approach to characterizing proteins in tandem mass spectrometry data that we will cover is *de novo* sequencing. The program that we will be using is called PEAKS [10] which implements what it calls auto *de novo* sequencing (as opposed to an older approach known as manual *de novo* sequencing which is largely done by hand). Other programs that also do auto *de novo* include PepNovo[11], Lutefisk[12], NovoHMM[13], and DirecTag[14].

In terms of algorithms, there are many different approaches. One naive approach is to list all possible candidates available based on the precursor ion mass and then comparing those with the fragment mass spectrum. The most popular implementation is to use a spectrum graph approach (as shown in Figure 5) in which peaks in the fragment mass spectrum form nodes and potential relationships between the nodes correspond to particular ions. The sequence that will be sought is a traversal of the graph that maximizes the scores on each of the relationships in the traversal. The traversal starts at vertices that correspond to the N and C termini.

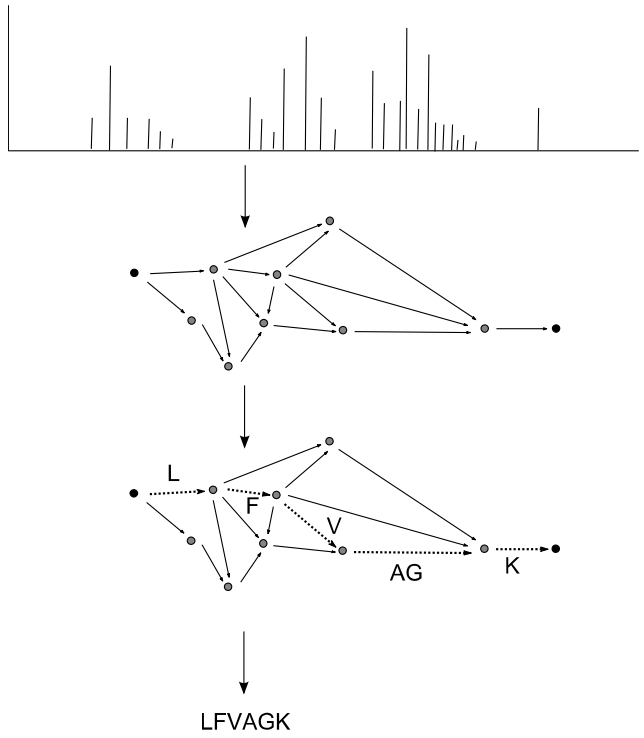

Figure 5: Spectrum graph approach

An alternative approach used in PEAKS is a reward/penalty score implemented via dynamic programming which rewards or penalizes the presence of peaks in the mass spectrum corresponding to candidates generated by the parent mass. Effectively, we simulate a mass spectrum for each possible candidate and reward peaks that match between the simulation and the real mass spectra and penalize peaks that do not match. Such an approach also has to take into account not only the dominant b and y ions (which are complimentary) but also has to take into account pairs of z and a ions as well as x and c ions which are also complementary respectively. The dynamic programming comes in when the algorithm attempts to grow from a partial sequence to cover the entire sequence.

We can describe one potential problem that will particularly affect our work with *de novo* sequencing and homology search. Consider Figure 6 which depicts an attempt at assigning a particular peptide sequence to mass spectrum. As we can see, the spectrum is noisy (many peaks are not assigned to any amino acids) and there are clusters of peaks (created by different isotopes incorporated). Additionally, we can see that the c-ion sequence lacks a peak to distinguish between TL and LT. If the z-ion sequence had been missing a peak to split T and L as well, then there would

be no way to determine the true sequence and we would be left with a 50% chance of choosing the wrong sequence. This kind of error is very common in *de novo* sequencing.

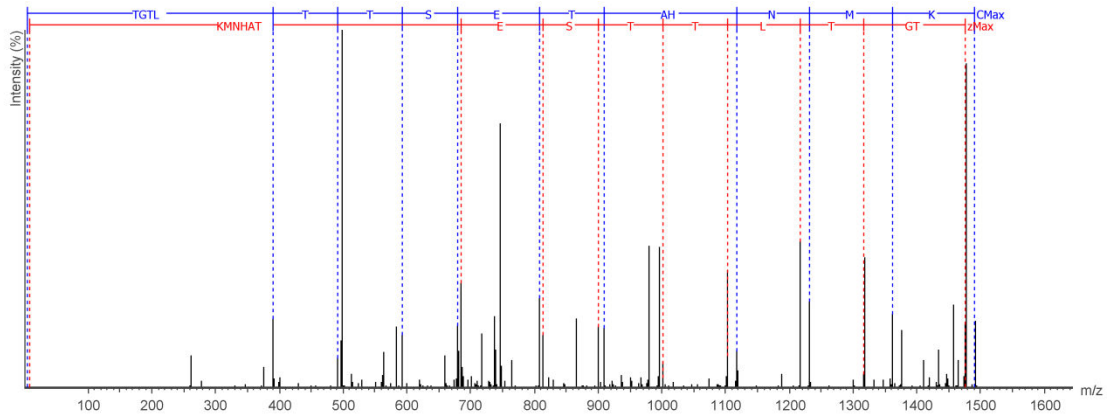

Figure 6: Sequence alignment

We can see an example of this actually happening with the first and second ranked candidates for the spectrum depicted in Figure 7 and Figure 8 respectively. Here, we can see that due to the lack of a peak between A and G, there is no way to distinguish between the two candidates that are identical aside from that segment and thus they both receive the same score.

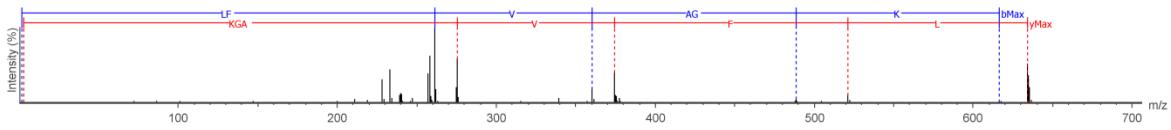

Figure 7: *De novo* sequence for LFVAGK

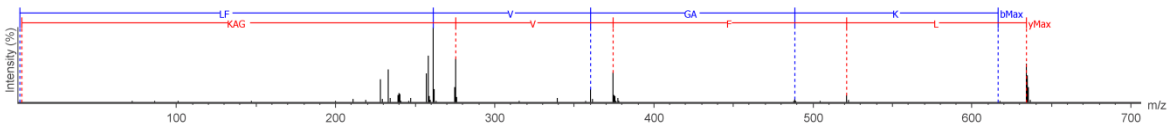

Figure 8: *De novo* sequence for LFVGAK

Alternatively, if we think of the sequence LFVAGK as a sequence of mass tags, we can see that since the peak between A and G was missing, we essentially have an ambiguous sequence of LFV[128.1]K since mass(AG) = mass(GA) = 128.1. Other examples of mass replacement errors in other sequences would be mass (WD) ≈ mass (VMA) and mass (RDG) ≈ mass (VTK). We can see further examples of this in Table 1 which displays all amino acids along with their masses and some

selected duplicate mass values. The number of duplicates increases rapidly as we look at larger and larger mass values.

Mass	Sequence	Name (For single AAs)
57.0215	G	Glycine
71.0371	A	Alanine
87.0320	S	Serine
97.0528	P	Proline
99.0684	V	Valine
101.0477	T	Threonine
103.0092	C	Cysteine
113.0841	I, L	Isoleucine, Leucine
114.0429	N, GG	Asparagine
115.0269	D	Aspartic Acid
128.0586	Q, AG	Glutamine
128.0950	K	Lysine
129.0426	E	Glutamic Acid
131.0405	M	Methionine
137.0589	H	Histidine
147.0684	F	Phenyalanine
156.1011	R	Arginine
158.0691	AS, GT	
160.0307	C(+57), CG	
163.0633	Y	Tyrosine
170.1055	AV, GL	
171.0644	NG, GGG	
186.0793	W	Tryptophan

Table 1: Mass Table (All AAs and selected duplicated mass values up to the mass of tryptophan)

2.4 Peptide Identification with Database Search

The second approach to characterizing proteins in tandem mass spectrometry data is called database search. Programs in the field that take this approach include PEAKS Protein ID[10], Omssa[16], Mascot [17], X!Tandem[18], and Sequest[19]. The database search approach actually pre-dates *de novo* sequencing and essentially reduces the scope of the *de novo* sequencing problem. By limiting possible sequence matches to those contained in existing protein databases, the scope of the problem is greatly reduced and more care/processing time can be taken to consider possible matches. The drawback is obvious. Only known protein sequences can be matched and any truly novel sequences will not be detected using this technique.

These algorithms in general, identify a matching peptide sequence in a database for each spectrum and then group these peptides together to identify the specific proteins. The first step often involves generating a theoretical MS/MS spectrum for each potential peptide in the database which is then compared with the actual spectrum. After the peptide sequencing, the originating proteins have to be identified. This can be a difficult task since proteins usually are homologous to many other proteins which can be difficult to choose between.

A *de novo* algorithm can handle PTMs by either adding additional amino acids as choices or modifying existing ones (called variable or fixed PTMs respectively). Since databases do not typically contain PTMs, one approach a database search algorithm can take is to consider variants of a database sequence with PTMs [20].

Chapter 3
Survey of Related Work

Previously, we covered two complementary approaches to characterizing proteins in mass spectrometry data, *de novo* sequencing and protein database search, both with different drawbacks, and the possibility of sequencing error versus the inability to identify peptides that are not previously sequenced. The way of achieving some kind of middle ground between these two approaches is to consider additional information via an understanding of evolution. Consider the following quote from Theodosius Dobzhansky [21]:

> Nothing makes sense in biology except in the light of evolution, *sub specie evolutionis*. If the living world has not arisen from common ancestors by means of an evolutionary process, then the fundamental unity of living things is a hoax and their diversity is a joke. The unity is understandable as a consequence of common descent and of universal necessities imposed by common materials.

Conversely, we can use an understanding of which organisms are homologous to one another (as in which organisms share common descent) to guide our characterization of mass spectrometry data and this is the approach used in a number of implementations of homology search.

3.1 Homology Search

On one hand, we have *de novo* sequencing algorithms that can directly create peptide sequences given the mass spectrum data but give no context. On the other hand, we have database search algorithms that can match mass spectrum data to known database peptides but cannot handle novel peptides.

The obvious bridge between these two approaches was to modify general homology search tools for the task of searching with (relatively) short sequences from *de novo* sequencing. Homology search tools in general answer the question whether there are any other genes or proteins that are related to the query sequence provided. Some of these tools were subsequently modified for MS/MS such as MS BLAST [22] as a derivative of the well-known general BLAST [23] algorithm.

The problem with this kind of approach is that these derivative tools were originally designed for much longer query sequences and do not take into account *de novo* sequencing error. As a simple example, a general homology search tool will heavily penalize a "mutation" in a sequence where two adjacent amino acids are swapped (which would look like an insertion followed by a deletion). However, a *de novo*-aware search tool will recognize that this is a common error resulting from a

missing peak as described in 2.3 and give a much lower penalty. In fact, certain errors such as the switching of isoleucine (I) and leucine (L) which have the same mass are impossible to resolve while the switching of lysine (K) and glutamine (Q) which have very similar masses cannot be distinguished except with high accuracy instruments (meaning greater than 0.03 Dalton accuracy). These errors can be completely ignored by a homology search tool which is *de novo* aware. This way, a genuine match will not be thrown off by a relatively common error, *de novo* sequencing errors being much more common than real mutations. It is this awareness which gives rise to Spider.

There are also a number of related approaches to Spider, which include TagRecon [24] which builds upon the work of DirecTag [14], and OpenSea [25]. In the sequence tagging approaches, a tag inference engine such as DirecTag creates a tag from a spectrum (which does not necessarily cover the entire spectrum and may have mass gaps). This tag filters candidate peptides that are obtained from theoretically digesting protein databases. It is worth noting that the combination of DirecTag and TagRecon is influenced by the older GutenTag [26] which produces short *de novo* sequences that are searched against a protein database, but without considering mutations or modifications. In particular, TagRecon identifies mutated peptides by considering various factors such as scoring based on number of reinforcing overlapping matches, the possibility of contaminants, a higher score for a mutation versus a non-mutant and enzyme information at the terminals.

Sequence tagging approaches (such as TagRecon) make it possible to allow a mass error on one side of the tag match and explain it by mutating residues. This approach is actually somewhat similar to Spider segment match which will be covered later. The challenges here are that sequence tags are often shorter than *de novo* sequences due to allowing mass gaps which makes discrimination more difficult since a shorter or less specific tag can potentially match more possible candidates. Additionally, there are trade-offs in terms of sensitivity and specificity when it comes to the lengths of tags and the number of tags kept.

Another alternative approach is to modify a database search algorithm from section 2.4. Note that this does bypass the need for a *de novo* sequence or a sequence tag. One implementation of the alternative is the Mascot error tolerant search [20]. This search attempts to compensate for a number of different failures that can lead Mascot to not return a match including systematic error, enzyme non-specificity, and post-translational modifications, in addition to the mutations that we are interested in. The search is implemented in two passes, a standard first pass in which the standard database search algorithm is used and a second pass in which a number of constraints are loosened

and additional possibilities considered, but only against a selected number of proteins that were identified in the first pass.

It is important to note that because the second pass considers such a large number of possible errors, it is not entirely comparable to Spider. It does, however, handle enzyme non-specificity by testing all possible sub-sequences of a possible database match; it handles precursor charge errors by trying multiple charge states, systemic error by calibrating via strong matches, and unsuspected modifications by testing against one modification at a time from a specific list. These problems are often handled in the *de novo* sequencing or tag matching steps of other approaches. The approach to handling mutations is by considering possible mutation errors by allowing for one underlying nucleic acid mutation and considering all the possible amino acids that could result. This is basically equivalent to a PAM10 matrix at 72% identity with at a threshold of -8.27[8].

Mascot's error-tolerant search is thus implemented in two versions, an automatic error tolerant search that chooses high-scoring peptides on which to base a second pass and a manual version in which the user will select proteins to be included in a second pass. In 4.4 we will use the first mode since it is recommended by its authors and in order to eliminate bias when choosing proteins for the second pass.

3.2 Previous Spider Work

Spider (Software Protein IDentifER) was initially developed in 2005[3] by Yonghua Han, Bin Ma, and Kaizhong Zhang. Additional details were provided in Yonghua Han's thesis [27]. It will be useful to go over some of the previous research and implementation work covered in these works before going over the improvements that will be addressed in this thesis.

First, a mathematical foundation for the matching task between *de novo* sequence and peptide sequence was created. Briefly, let X, Y, and Z be the de novo sequence, the real sequence, and the database sequence, respectively. Given the function f which denotes the *de novo* sequencing error between two sequences and g which denotes the edit distance or mutation distance between two sequences, we can think of our general problem as determining the following score:

$$d(X, Z) = \min_y (f(X, Y) + g(Y, Z)) \quad (3.1)$$

[8] PAM10 is a type of alignment matrix that scores potential pairs of amino acids based on their likelihood of being the result of mutation from one another.

Note that *Y* can be different from both *X* and *Z* since both *de novo* error and mutations respectively can occur. Furthermore, since the algorithm does not know *Y* at the beginning of the search, we have to find a sequence of potential *de novo* errors and homology mutations that minimizes both.

Therefore, the core problem is really to find this sequence *Y* and we can think of the sequence tag search problem in terms of minimizing the distance across all possible matches. Once *Y* is found, the distance functions can be computed easily and *Y* is also likely to be the real peptide sequence since it is similar to a real database peptide *Z* and a partially correct sequence tag *X* computed by *de novo* sequencing. In Figure 9 we see that we can think of an optimal alignment of the three sequences as a merger between an alignment between (*X*, *Y*) and (*Y*, *Z*). In a) we see that mass segment errors can occur in one column whereas b) demonstrates a mutation (substitutions/insertions/deletions) which can lead to our definition of "blocks."

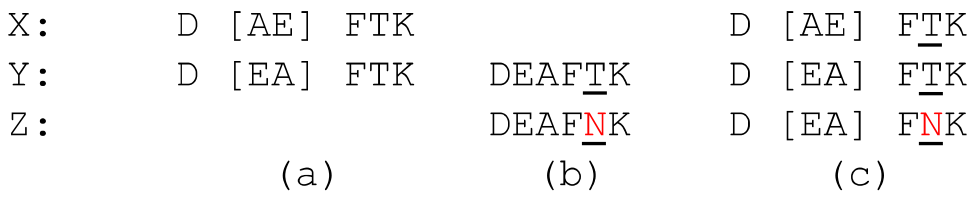

Figure 9: Example of block-wise alignment for specific *X*, *Y*, and *Z*

Keeping in mind this concept of "blocks", we can see that we can consider each block in isolation when calculating the solution to the general problem $d(X, Z)$.

$$\begin{aligned} d(X,Z) &= \sum_i d(X_i, Z_i) \\ &= \sum_i \min_y(f(X_i, Y_i)) + \sum_i \min_y(g(Y_i, Z_i)) \\ &= \sum_i \min_y\big(f(X_i, Y_i) + g(Y_i, Z_i)\big) \end{aligned} \tag{3.2}$$

When considering the cost of a particular block, we will need to consider a related cost, the cost of matching a particular block from a *de novo* sequence with a block from a protein sequence when there is sequencing error. We will define this as $\alpha(X, Z)$. However, due to the fact that *de novo* sequencing actually can be considered to return a mass segment when there is a sequencing error, we will in fact use a different cost, $\beta(m, Z)$ where m is defined in terms of mass. The relationship between the two is shown below.

$$\begin{aligned}\alpha(X,Z) &= f(m) + min_{Y:m(Y)=m}\, g(Y,Z) \\ &= f(m) + \beta(m,Z)\end{aligned} \tag{3.3}$$

In other words, when given a possible match, we need to determine a real block Y which minimizes the cost g between Y and Z (in reality, we will also need to consider the cost of a mass tolerance error when $m(X) \rightarrow m(X) \pm \delta$. δ would be a value representing the possible imprecision of the underlying instrument in determining the position of peaks).

For the alignment cost of $\beta(m,Z)$ as shown in Figure 10 there are three cases of the alignment. These cases involve the chance of a deletion in *Z*, a deletion in *Y*, or the chance of an amino acid match.

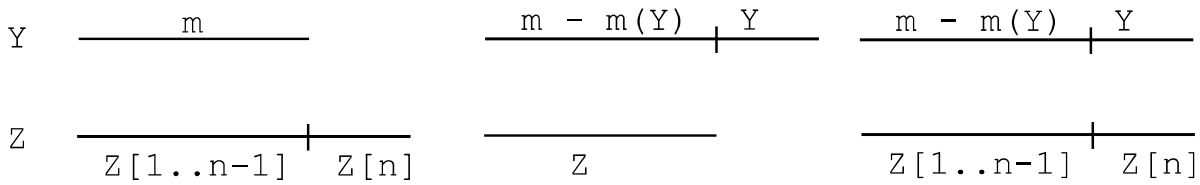

Figure 10: Cases for β(m,Z)

For the chance of an amino acid match, we will require one more building block of the Spider algorithm. This is the series of block substitution matrix (BLOSUM) [28]. These matrices were created by examining over 500 groups of multiple sequence alignments between proteins that were distantly related to one another. By keeping track of how aligned amino acids vary from alignment to alignment, it is possible to establish substitution frequencies which are also grouped by the degree of similarity that each aligned block displays. In this way, different matrices such as a BLOSUM62 or BLOSUM90 matrix can be created. The number indicates that it was created from proteins with less than 62% identity or less than 90% identity respectively. Spider uses a BLOSUM90 matrix with the 90 reflecting the fact that we aim to work on proteins with little divergence.

With this in mind, we can develop the dynamic programming algorithm that follows:

Algorithm for computing $\beta(m, Z[i..j])$

Where $dist(y, z)$ refers to the cost of a BLOSUM alignment
Where *indel* refers to the cost of a gap penalty in the case of an insertion or deletion

1. FOR m from 0 to $m(X)$ step Δ $\quad //\sim O(m \times |Z|^2 \times |\sum|)$
2. FOR i from 0 to $|Z|$ $\quad //\sim O(|Z| \times |Z| \times |\sum|)$
3. FOR j from 0 to $|Z|$ $\quad //\sim O(|Z| \times |\sum|)$
4. $$\beta(m, Z[i..j]) = min \begin{cases} \min_y \beta(m - m(y), Z[i..j]) + indel \\ \beta(m, Z[i..j]) + indel \\ \min_y \beta(m - m(y), Z[i..(j-1)]) + dist(y, Z[j]) \end{cases} \quad // \sim O(|\sum|)$$

The time complexity of this algorithm thus works out to $O(m \times |Z|^2 \times |\sum|)$ where $|\sum|$ represents roughly the number of amino acids (and the number of alternative amino acids due to the possibility of PTMs). Continuing on, we will go back to considering the full cost of computing $D(X, Z)$. For the full cost of $D(X, Z)$ we can do a similar analysis with four cases including the case where β cost comes into play. Figure 11 demonstrates these cases as when there is a deletion in *Z*, when there is no *de novo* error but *y* is removed in a match with *x*, when there is no *de novo* error and *y* is removed in a match with both *x* and *y*, and finally, when there is *de novo* error involving both *x* and *z*.

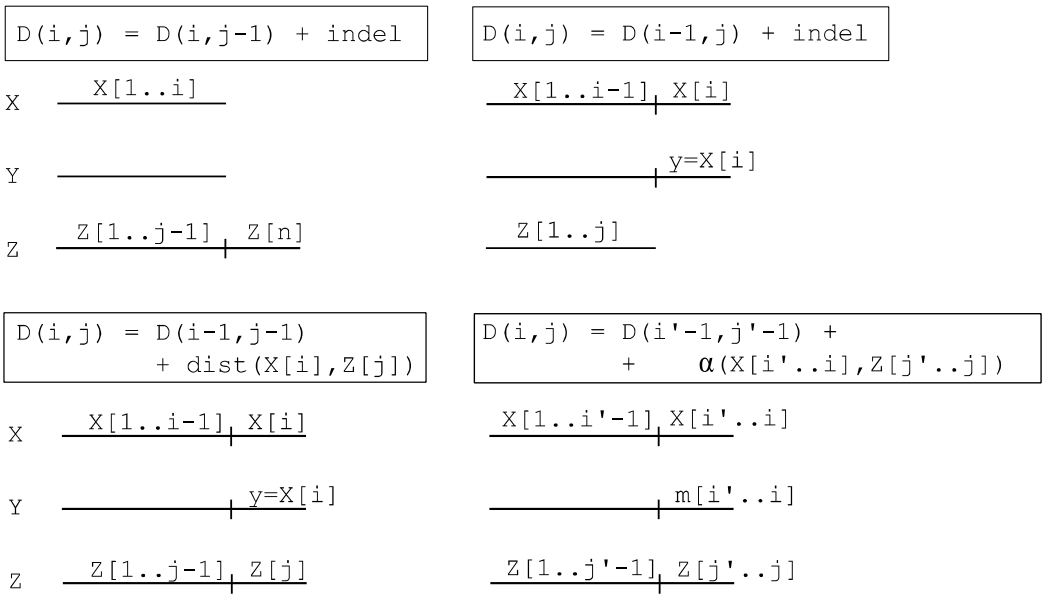

Figure 11: Cases for the Cost of $D(X, Z)$

These cases in turn give rise to the dynamic programming algorithm which can be used for search and to reconstruct the real sequence *Y*.

Algorithm for computing $d(X, Z)$ and Y

1. FOR i from 1 to $|X|$ $//\sim O(|X|^2 \times |Z|^2)$
2. FOR j from 1 to $|Z|$ $//\sim O((|X| \times |Z|) \times |Z|)$
3. $$D(i,j) = min \begin{cases} D(i-1,j) + indel \\ D(i,j-1) + indel \\ D(i-1,j-1) + dist(X[i], Z[j]) \\ \min_{i',j'} D(i'-1,j'-1) + \beta(m(X[i'..i], Z[j'..j]) + f(m(X[i'..i])) \end{cases}$$ $//\sim O(|X| \times |Z|)$

After the algorithm is run, we can simply output $D(|X|, |Z|)$ as $d(X, Z)$ which gives us a score for this particular match between a given *de novo* sequence and a potential identification for a peptide from a homologous protein. Additionally, if during the algorithm, the block y_i chosen in each step is recorded, we can use a backtracking algorithm in order to reconstruct the entire middle sequence Y.

Combining the algorithm for computing $d(X, Z)$and Y with the previous algorithm for computing $\beta(m, Z[i..j])$ we get a total runtime of:

$$\begin{aligned} Total\ cost &= O(m \times |X|^2 \times |\sum|) + O(|X|^2 \times |Z|^2) \\ &= O(|Z|^4 + |Z|^2 \times m \times |\sum|) \end{aligned} \quad (3.4)$$

A straightforward naïve implementation of the Spider algorithm while polynomial in order and feasible for reconstruction would not be practical for searching. The reason for this difference is that reconstruction requires one run of the algorithm for each final match versus searching which requires one run of the algorithm for each possible match. An implementation of the naïve algorithm resulted in performance of roughly 5 *de novo*-peptide matches per second resulting in a runtime of over 14 days over a dataset of 412 spectra due to the number of possible *de novo*-peptide matches. A more sophisticated algorithm took roughly 628 seconds over the same dataset in 4.3.2.

As a result, Spider implemented a number of heuristics and search modes in order to trade-off the complexity of the model against increased run-time. Users can pick different search modes depending on their needs. In the implementation, these heuristics come into play in order to evaluate possible matches after a hashing algorithm was used to generate potential matches from the query sequences by using 3-mer amino acid seeds. These seeds are used as entry points into the protein database and for each possible seed, one of the following Spider heuristic match modes was used to further evaluate the match and determine its score:

3.2.1 Exact Match

This heuristic assumes that the *de novo* sequencing result has no errors beyond leucine and isoleucine or lysine and glutamine substitution. It also assumes that there is no mutation in the database. This was a fast reference mode for Spider and was not used in practice for homology search. However, this is still useful for tag matching as a different approach for database search in Peaks DB [10].

3.2.2 Segment Match

Segment match allowed for *de novo* sequencing errors up to three amino acids but no mutations in the database. This mode acts as a quick search mode (linear per match) that allows for searching a database given *de novo* results, but is not a true homology match.

```
De Novo:    W  [WE]  QVVLAMPYDT  [VP]  [MW]  R
            |        ||||||||||              |
Database:   W [VDT]  QVVLAMPTDY [PV]  [PGY]  R
```

Figure 12: Example of Segment Match

An example is given in Figure 12, with *de novo* sequencing errors marked with square brackets. As we can see, mass (WE) =315.122 ≈ mass (VDT) = 315.143 while mass (VP) = mass (PV) and mass (MW) = 513.241 ≈ mass (PGY) = 317.138.

3.2.3 Non-gapped Homology Match Mode

This search mode allows for substitution mutations which do not exist in the same block as a *de novo* error. This is in addition to the cases covered by the previous search mode. A greedy heuristic is used to find a good alignment by extending up to three amino acids from an existing match and recording local maximum scores while extending. An example is shown in Figure 13 which demonstrates two locations with a substitution mutation, but no *de novo* errors.

```
De Novo:    T <ME>  ADESHA <DC>  EK
            |       ||||||       ||
Real:       T <ME>  ADESHA <DC>  EK
            |       ||||||       ||
Database:   T <CV>  ADESHA <GC>  EK
```

Figure 13: Example of Non-gapped Homology Match

3.2.4 Gapped Homology Search Mode

This search mode was the most complex and most thorough. As a heuristic, Spider used what was called a S4 matrix to determine what were deemed "bad-blocks" and "good-blocks." This way, Spider could consider mutations that may occur in the same block as a *de novo* error. Figure 14 demonstrates the possibility of a *de novo* error occurring in the same block as a mutation since PY and FI have approximately the same mass. FI also is the most likely mutated block corresponding to CV with that mass.

```
De Novo:    GTGQE <PY> PNSNER
            ||||| |  | ||||||
Real:       GTGKE <FI> PNSNER
            ||||| |  | ||||||
Database:   GTGKE <CV> ONSNER
```

Figure 14: Example of Gapped Homology Match

In terms of implementation details, Spider was implemented as a server-based Java application.

3.2.5 Protein Re-sequencing

This section will survey other instances of protein re-sequencing. Here, we will define protein re-sequencing as any approach that allows for the complete sequencing of a novel protein from MS/MS data.

Protein sequencing with Spider is possible due to some of the developments that have been covered in the previous sections. First, *de novo* sequencing has progressed to the point where it can compute highly accurate sequences of reasonable length. Second, database search (or homology search) algorithms can identify a likely candidate which is highly related to the novel protein in question and can be used for sequencing as a base. Third, Spider can be used to align *de novo* tags with the reference protein and reconstruct likely tags that more accurately characterize the real sequence. Finally, on the biological side, it is possible to use multiple overlapping peptides in order to align peptides to the extent that there is a complete non-broken series of Spider tags which characterize the full protein. This is the basic outline of how Spider and Champs will go about protein re-sequencing, but there have also been other approaches in the past and present that have tackled the problem in different ways.

One early example of determining the protein sequence via mass spectrometry is one approach in 1989 on Glutaredoxin [29]. Two different enzymes (Trypsin and thermolysin) were used to create overlapping peptide sequences obtained by manual *de novo* sequencing. These were arranged together by the sections that were overlapping in order to yield the complete protein sequence. A similar approach was used to characterize carnocyclin A [5], a circular bacteriocin. However, in this case, manual *de* novo sequencing was verified with automated *de novo* sequencing in order to sequence peptides. These peptides were then assembled using their overlapping ends.

However, the current aim is to yield actual programs that can be used for multiple proteins quickly. Two examples include an automated software tool that achieves 96% coverage with 90% accuracy [30] and a subsequent approach that improved coverage to 97% to 99% but without discussing accuracy [31]. In the first approach, called "shotgun protein sequencing" by its authors, multiple spectra are assembled using a technique called spectral alignment. These assembled spectra are subsequently *de novo* sequenced. The second approach adds known proteins as templates when considering the problem on monoclonal antibodies. Its workflow consists of a three-stage approach involving alignment, assembly, and consensus. Again, the alignment stage is performed via spectral alignment into "spectral contigs" while the assembly stage assembles these spectral contigs into "protein contigs" which can be subsequently analyzed.

One other competing approach is called "template proteogenomics [32]," which relies upon the detection of anchors and their extension using HMM (Hidden Markov Models) models of spectra that overlap with these anchors. This approach actually follows the first publication of Champs and differs in some important ways. In their case, the work focused on re-sequencing highly variable immunoglobulins (antibodies). The high amount of variation was due to hyper-mutation owing to a large number of possible antigens.

Their algorithm used a database search tool (InsPecT [33]) to identify possible anchors (and thus was vulnerable to mutated anchors). Anchor extension was done by recruiting overlapping spectra that could overlap both the anchor and part of the template[9]. In this way, the program could train a HMM to generate the exposed spectra on the section extending out from the anchor. This allowed the program to build a consensus sequence which could then be fed into a spectrum-graph based *de novo* algorithm in order to build a full sequence.

[9] The term template was used as their terminology for a homologous protein

Chapter 4
Spider II

4.1 Theory

As detailed in 3.2, the initial implementation of Spider resulted in an algorithm that was theoretically $O(|Z|^4 + |Z|^2 \times m \times |\sum|)$ in runtime. However, a straightforward implementation would be too slow and thus it was modeled in a number of heuristic algorithms. These heuristic algorithms vary based on whether they considered mutations and mutations overlapping with *de novo* sequencing errors. Thus, while there was an in-depth mathematical model for the algorithm, the actual scoring function used in Spider was largely empirical and much simpler. For example, the probability that an amino acid was correctly assigned was set to 80%, the gap penalty was set to 10, and many of the penalties associated with the S4 scoring matrix were empirically set. Therefore, some new ideas were considered in order to give a theoretical basis to the scoring function.

In this section, we propose an upgrade to the algorithm that should result in a more accurate and efficient implementation which also downgrades the emphasis on heuristics in favour of a more rigorous algorithm. The upgrade that we propose will also improve performance by removing the dependence on the number of amino acids, denoted as $|\sum|$, at runtime. Last, but not least, the final iteration of the new search algorithm will unify search with reconstruction ensuring that the reconstructed sequence reflects the sequence that the search algorithm implicitly calculated.

Again, we let X, Y, and Z be the *de novo* sequence, the real sequence, and the database sequence, respectively. An alignment is defined by a series of blocks (X_1, Y_1, Z_1), …, (X_k, Y_k, Z_k). When a block involves only one letter for X and Y, *de novo* sequencing made a correct prediction on X in this block. Otherwise, the block involves a *de novo* sequencing error. We will also add the limitation that the blocks of X and Z have at most three amino acids each. This is good enough for most practical cases. This is due to the fact that when X_i contains more than three amino acids, the large number of same-mass combinations[10] makes X_i almost useless in terms of providing useful information. In such a case, we will arbitrarily separate X_i into several smaller blocks which will still be a usable approximation of having one large block.

[10] The start of this trend can be examined in the table of mass values and amino acids as shown in Table 1

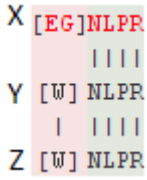

Figure 15: Alignment with five blocks

4.1.1 Alignment Score

We will define the alignment score as follows. We will start with $\frac{P(xyz)}{P(x)P(z)}$ which is the ratio of observing x, y, and z together in an alignment such as Figure 15 and the probability of observing x and z independently. If y can be introduced in the middle to make the alignment x, y, and z together very likely then z is likely to be the correct sequence.

$$
\begin{aligned}
f(x,z) &= \max_y \log \frac{P(xyz)}{P(x)P(z)} \\
&= \max_y \log \frac{P(x|yz)\,P(yz)}{P(x)P(z)} \\
&= \max_y \log \frac{P(x|y)\,P(yz)}{P(x)P(z)} \text{ by independence of x and z} \\
&= \max_y \log \frac{P(y|x)\,P(yz)}{P(y)P(z)} \text{ by Bayes' theorem} \\
&= \max_y \left(\log P(y|x) + \log \frac{P(yz)}{P(y)P(z)} \right)
\end{aligned}
\tag{4.1}
$$

However, while this definition is mathematically correct. It is quite unwieldy since it is defined across the entire length of the match between the two sequences. Therefore, we will take inspiration from the preceding discussion on blocks and we will define a score function to evaluate the quality of an alignment when it is split up into smaller blocks. The score will be analogous to a sequence alignment score using BLOSUM matrices. For each block, we will define the score to be

$$
sc(X_i, Y_{i,} Z_i) = \log \frac{P(X_i Y_{i,} Z_i)}{P(X_i)P(Z_i)} \tag{4.2}
$$

Then we define

$$
sc(X, Y, Z) = \sum sc(X_i, Y_{i,} Z_i) \tag{4.3}
$$

and

$$sc(X,Z) = \max_Y(sc(X,Y,Z)) \tag{4.4}$$

We can see that when (X_1,Y_1,Z_1), …, (X_k,Y_k,Z_k) is an optimal alignment, then

$$sc(X,Z) = \sum_i f(X_i,Z_i) \tag{4.5}$$

In other words, a score defined as the sum of alignment scores block by block is adequate for our needs. This gives us a definition for the alignment score which reflects the discussion on blocks in Section 3.2.

4.1.2 How to calculate $f(x,z)$

The second term of the definition $\log\frac{P(yz)}{P(y)P(z)}$ is the alignment score between y and z using a BLOSUM [28] matrix. Recall the definition of BLOSUM in 3.2.4 as a substitution matrix used for the alignment of peptide sequences and computed from multiple sequence alignments of conserved protein families. We will use BLOSUM90 as an approximation and denote this as $al(x,y) = \log\frac{p(yz)}{p(y)p(z)}$. This also means that all log calculations will be in base 2 (log_2) in order to match the base of the BLOSUM scores. Additionally, since the BLOSUM scores are calculated with a scaling factor of 2 in order to match an older type of alignment scores, we must also halve alignment scores in practice.

The first term $\log\ P(y|x)$ has a few cases:

Case 1: $|m(y) - m(x)| > \delta$. This means that the real sequence is out of the mass tolerance δ of the *de* novo sequence which is impossible. Therefore, $P(y|x) = 0$, $\log P(y|x) = -\infty$, which corresponds to the probability that x cannot be the de novo sequencing result of y. Therefore

$$f(x,z) = -\infty \tag{4.6}$$

Case 2: $y = x$. If the *de novo* sequencing is correct, then both x and y are one letter long and $P(y|x) = 1$ then $\log\ P(y|x) = 0$. Therefore combining the two terms from (4.1), we get

$$f(x,z) = al(y,z) \tag{4.7}$$

Note that in practice, while x and y are only one letter long, in practice z may be longer (or even of length 0). In this case, we will use a gap penalty to compute the alignment score.

Case 3: $|m(y) - m(x)| \leq \delta\ and\ y \neq x$

This means that we have a sequencing error since the *de* novo sequence does not match the real sequence. We define the probability $p(m)$ as the sum of the probability of all the different blocks that result in approximately the same mass m.

$$P(m) = \sum_{m(u)\approx m} P(u) \tag{4.8}$$

We will define $c(a)$as the *de novo* sequencing confidence on amino acid a. In case 3, we can adjust the probability of a mass gap by the confidence score. In the case where the sequencing algorithm gave a very confident prediction, case 3 is very unlikely. However, if the algorithm is not particularly confident due to, for example, missing peaks, then case 3 is much more likely. Therefore, we will define $w(x)$ as the multiplication of $(1 - c(a))$ for every amino acid *a* in the block x, meaning the probability that we get every *de novo* sequenced amino acid wrong.

$$P(y|x) = \frac{w(x) \times P(y)}{P\big(m(x)\big)} \tag{4.9}$$

Therefore

$$\log P(y|x) = \log w(x) - \log\Big(P\big(m(x)\big)\Big) + \log P(y) \tag{4.10}$$

Thus combining the two terms of $f(x,z)$ again, we get

$$f(x,z) = \log w(x) - \log\Big(P\big(m(x)\big)\Big) + max_{m(y)\approx m(x);y\neq x}(\log P(y) + al(y,z)) \tag{4.11}$$

Combining the equations (4.6), (4.7), and (4.11), we have

$$f(x,z) = \max\begin{cases} al(x,z) \\ log\, w(x) - \log\Big(P\big(m(x)\big)\Big) + \max_{m(y)\approx m(x);y\neq x}(\log P(y) + al(y,z)) \end{cases} \tag{4.12}$$

4.1.3 How to calculate an alternate $f(x,z)$

It should also be noted that during the course of our research we also derived the following equation for $f(x,z)$. In this alternative formulation, we defined the alignment score as follows:

$$\begin{aligned} f(x,z) &= \max_y \log \frac{P(xyz)}{P(x)P(y)P(z)} \\ &= \max_y \log \frac{P(x|yz)\,P(yz)}{P(x)P(y)P(z)} \\ &= \max_y \log \frac{P(x|y)\,P(yz)}{P(x)P(y)P(z)} \text{ by independence of x and z} \\ &= \max_y \log \frac{P(y|x)\,P(yz)}{P(y)P(y)P(z)} \text{ by Bayes' theorem} \\ &= \max_y \left(\log \frac{P(y|x)}{P(y)} + \log \frac{P(yz)}{P(y)P(z)}\right) \end{aligned} \tag{4.13}$$

Note the added term (y) . In the previous formulation, we considered that y was constructed from x and z making it non-random and thus unfair to divide by it. However, this derivation may also be valid.

As before, the second term of the definition is unchanged with $\log\frac{P(yz)}{P(y)P(z)}$ forming the alignment score between y and z and denoted as $al(x,y) = \log\frac{p(yz)}{p(y)p(z)}$. We will work through the changed cases of the first term.

Case 1: $|m(y) - m(x)| > \delta$. Then $P(y|x) = 0$, $\log\frac{P(y|x)}{P(y)} = -\infty$, which means x cannot be the de novo sequencing result of y. Therefore this case remains unchanged

$$f(x,z) = -\infty \tag{4.14}$$

Case 2: $y = x$. Then $\log\frac{P(y|x)}{P(y)} = -\log\big(P(x)\big)$. Therefore this case is now

$$f(x,z) = -\log\big(P(x)\big) + al(y,z) \tag{4.15}$$

Case 3: $|m(y) - m(x)| \leq \delta \; and \; y \neq x$

As before, we arrive at

$$P(y|x) = \frac{w(x) \times P(y)}{P\big(m(x)\big)} \tag{4.16}$$

However

$$\log\frac{P(y|x)}{P(y)} = \log w(x) - \log\Big(P\big(m(x)\big)\Big) \tag{4.17}$$

Thus

$$f(x,z) = \log w(x) - \log\Big(p\big(m(x)\big)\Big) + \max_{\mathrm{m(y)\approx m(x);y\neq x}}(al(y,z)) \tag{4.18}$$

Combining the new equations (4.14), (4.15), and (4.18) , we would now have

$$f(x,z) = \max\begin{cases} -\log P(X) + al(x,z) \\ \log w(x) - \log\Big(P\big(m(x)\big)\Big) + \max_{\mathrm{m(y)\approx m(x);y\neq x}}(al(y,z)) \end{cases} \tag{4.19}$$

However, while this alternate formula is also mathematically valid, when this alternate formula was put through the testing that we will present in 4.3, we found that the results were less than desirable. In brief, while performance in terms of correct amino acids and correct matches was comparable to results derived from the previous formula, we found that this formula is significantly less capable of separating true positives from false positives. The empirical reason is that the term $-\log P(X)$ adds

roughly the same amount to every amino acid in a potential match varying only by the prevalence of a particular amino acid in protein databases[11]. This skews the score toward favouring longer matches versus matches with good alignment score. Therefore, it was decided to use the previous formula for $f(x,z)$ for our final work.

4.1.4 Recurrence Relation

We will combine the following recurrence relation with the final alignment score in order to compute the optimal alignment.

$$sc(X[1..i], Z[1..j]) = \max_{i-3 \le i' < i;\, j-3 \le j' \le j} \left(sc(X[1..i'], Z[1..j']) + f(X[i'+1..i], Z[j'+1..j]) \right) \quad (3.5)$$

Let $(X[1..i], Z[1..j])$ be the optimal alignment that results in the maximum alignment score. By the definition of our score, when we have an optimal alignment then the alignment score is equal to the sum of alignment scores calculated block-by-block. We will consider alignments involving a prefix of $(X[1..i'], Z[1..j'])$ and an alignment score for $(X[i'+1..i], Z[j'+1..j])$.

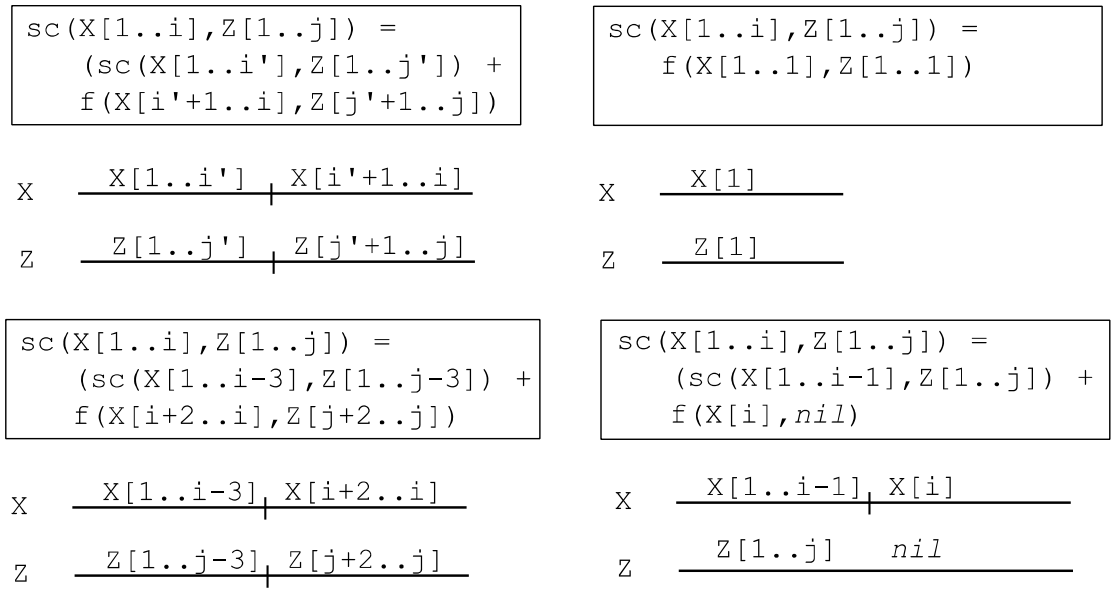

Figure 16: Recurrence Relation with selected cases

[11] We used the NCBI nr database to calculate frequency data

In order to prove this recurrence relation, we will consider the possible cases while examining the Figure 16 which demonstrates some of the possibilities. Going clockwise from the upper right, in the trivial base case there is no prefix and we only have to consider the alignment provided by the final alignment score. In the base case, $f(x,z)$ calculates the maximum score and there is only one possible alignment when there is only one amino acid in both parts of the alignment. Moving to the lower right, we examine one more border case when the new block is the smallest. We have one amino acid aligned with a gap. This corresponds to Case 2 of $f(x,z)$. Moving to the lower left, we see the other extreme when the new block involves an alignment between the largest possible (≤ 3)-mers. Finally, moving to the upper left, we see the general case. By examining the range of the indexes$\left(\mathrm{i}-3\leq \mathrm{i}'<i; j-3\leq \mathrm{j}'\leq \mathrm{j}\right)$, we see that all possible (≤ 3)-mers are considered for the last block (which falls between the last two cases in terms of size), and the max operation means that the optimal alignment which corresponds to the maximum alignment score will dominate.

This highlights an important consequence of our decision to only consider (≤ 3)-mers. This allows us to only consider a small upper bounded set of possible blocks (12 combinations of i' and j') within a small region near the last block in order to construct the optimal alignment.

For the implementation, along with the BLOSUM table for all (≤ 3)-mers[12], we will need to pre-calculate:

1. $\log\left(P\left(m(x)\right)\right)$ for all (≤ 3)-mers x
2. $\max_{m(y)\approx m(x);y\neq x}(\log P(y)+al(y,z))$ for all (≤ 3)-mer pairs x and z.

Finally, there is one note that will improve the resulting algorithms. If the confidence of amino acid sequencing is lower-bounded by a small constant (perhaps $\frac{1}{20}$ as the output of a *de novo* sequencing cannot be worse than a random sequence), then it is probably fine in theory to remove $y\neq x$ from $\max_{\mathrm{m}(y)\approx \mathrm{m}(x);y\neq x} al(y,z)$. In practice, case 2 will take over the maximisation in the cases where $x=y$ in case 3 making it a moot point.

So, instead of pre-calculating $max_{\mathrm{m(y)}\approx \mathrm{m(x)};\mathrm{y}\neq \mathrm{x}}(logP(y)+\ al(y,z))$ for all (≤ 3)-mer pairs x and z, we can pre-calculate $max_{\mathrm{m(y)}}\ al(y,z)$ for all pairs of mass value m and (≤ 3)-mer z. This is both faster and more memory-efficient. This was in fact used in the implementation of Spider II.

[12] (≤3)-mers refer to all blocks with either one, two, or three amino acids in them

4.1.5 Algorithms

We will next cover the various parts of the algorithms that are necessary to implement the theory introduced in 4.1. As in the initial Spider paper, many of the algorithms are implemented using dynamic programming. In dynamic programming, problems are expressed in terms of smaller sub-problems, as in recursion. The results of calculating these sub-problems are stored, which saves computation time since these results are later used again.

Typically, in sequence alignment problems such as Spider, after formulating the problem recursively using our recurrence relations, we can lay out the terms of the sequence in the form of a matrix which stores sub-problems to save computation time. This matrix also allows us to traceback or record the path taken through the matrix in order to recover the sequence of steps that was taken.

-	-	A	T	C	G	A	C
-	0	-4	-8	-12	-16	-20	-24
C	-4	-3	-7	-3	-7	-11	-15
A	-8	1	-3	-7	-6	-2	-6
T	-12	-3	6	2	-2	-6	-5
A	-16	-7	2	3	-1	3	-1
C	-20	-11	-2	-1	0	-1	8

Figure 17: Example of dynamic programming in Needleman Wunsch [34]

Specifically in the case of Spider, the first calculation is required to calculate Case 2 as outlined in theory. Case 2 is separated into two parts, a pre-calculation step that uses the Needleman-Wunsch[35] algorithm to calculate the alignment cost between all possible pairs of blocks and a quick runtime step that essentially involves a simple table lookup in the cache that is created by the first step. It should be noted that Needleman-Wunsch is itself a dynamic programming algorithm that uses a matrix as demonstrated by Figure 17.

In practice, all blocks of length 1 to 3 are coded with integers both as indexes into the cache and in terms of their mass. This simplification can be used to save memory since the fractional part of the amino acid masses is always insufficient to sum over 1 Dalton across all choices of three amino acids.

Algorithm: Case 2 - *Calculate* $al(x, z)$

Input: The choice of amino acids $\sum$
Output: Cache containing alignment score $al(x, z)$ for all pairs of blocks with length 1..3

Pre-calculation step:

Where NW is the Needleman-Wunsch algorithm

1. FOR(∀ block i from blocks of length 1..3) DO // $\sim O(|\sum|^6)$
2. FOR(∀ block j from blocks of length 1..3 < block i) DO // $\sim O(|\sum|^3)$
3. lookupCacheMap(i,j) ← NW(i,j)

Runtime step:

Input: One peptide block and one protein block
Output: Score for the match

1. X = peptide block
2. Z = protein block
3. return lookupCacheMap(X,Z) // $\sim O(1)$

The cost of the pre-calculation step is equal to iterating through all possible blocks of length 1 to 3 $O(|\sum| + |\sum|^2 + |\sum|^3)$ and performing one iteration of Needleman-Wunsch on each pair of blocks. Therefore we see that the pre-calculation step is a non-trivial $O(|\sum|^6)$ runtime. However, this can be totally separated from the runtime calculation which is $O(1)$.

The second calculation is Case 3. Case 3 is also separated into three parts, a pre-calculation step that also uses the lookup table resource from Case 2, and a more involved runtime step that still runs in constant time. The pre-calculation step will involve $\log(p(m(x)))$ for all (≤ 3)-mers which represents the log probability that we chose a combination of amino acids with the same mass as a particular n-mer and $\max_{m(y)\approx m(x);y\neq x}(\log P(y) + al(y, z))$ for all (≤ 3)-mer pairs x and z. However, note that we simplify this as hypothesized in the previous section to all possible pairs of mass values m and (≤ 3)-mers z giving a real calculation of $\max_{m(y)\approx m(x)}(\log P(y) + al(y, z))$.

Algorithm: Case 3 – Pre-calculation steps

Pre-calculate $\log(P(m(x)))$

Input: The choice of amino acids $\sum$
Output: Cache containing $\log(P(m(x)))$ for all blocks with length 1..3

Where freq(block) calculates the probability that we see a block in the full NCBI nr database

```
1.  FOR(∀ block i from blocks of length 1..3) DO                    // ~O(|∑|^6)
2.    x_blocks ← ∀ blocks where m(block) ≈ m(i) based on mass tolerance
3.    score ← 0
4.    FOR(block x : x_blocks)                                      // ~O(|∑|^3)
5.        score ← freq(x) + score
6.    cache3a(i) ← score
```

Pre-calculate $max_{\mathrm{m(y)}\approx\mathrm{m}}(logP(y) + al(y,z))$

Input: The choice of amino acids $\sum$
Output: Cache containing $\max_{\mathrm{m(y)}\approx\mathrm{m}}(logP(y) + al(y,z))$ for all pairs of possible block masses with blocks with length 1..3

```
1.  FOR(∀ blocks of length 1..3 with unique mass i  ) DO            // ~O(|∑|^9)
2.    z_blocks ← ∀ blocks where m(block) ≈ m(i) based on mass tolerance
3.    FOR(∀ block y from blocks of length 1..3) DO                 // ~O(|∑|^6)
4.      FOR(block z : z_blocks)                                    // ~O(|∑|^3)
5.        score ← freq(x) + lookupCacheMap(y,z)
6.        IF (score > maxScore)
7.          cache3b(i,z) ← maxScore
```

The proof of the pre-calculation runtimes is similar to the previous calculation. Each loop through all blocks of length 1 to 3 requires $O(|\sum| + |\sum|^2 + |\sum|^3)$. When iterating through all the potential blocks for a given mass, we have a worst cost runtime of the same. Each nested loop multiplies these resulting in the stated worst case runtimes of $O(|\sum|^6)$ and $(|\sum|^9)$ respectively. In practice, since most choices of amino acids do not result in many blocks of the same mass, the runtimes are much lower.

Again, these pre-calculation steps can be isolated from the actual runtime calculations. It should be emphasized that in these three cases, the runtime is really dependent on $|\sum|$ which corresponds to the number of amino acids under consideration and which is independent of the *de novo* sequences and the protein sequences that we will need to consider.

However, one consequence is that runtime changes depending on the number of PTMs under consideration as does the amount of memory required to store these caches. In the case of fixed PTMs, the cache does change due to the change in mass but the number of entries remains very similar. In the case of variable PTMs, the cache size grows rapidly since each modified amino acid is treated as an additional amino acid by the algorithm.

For example, the normal set of twenty amino acids results in a cache of 41.3MB. Adding three amino acids for Oxidation grows the cache to 46.6MB (12.8% growth) while adding seven amino acids for phosphorylation grows the cache to 62.2MB (50.6% growth).

Algorithm: Case 3 – Runtime Step

Input: One peptide block X with positional confidence, one protein block Z
Output: Score for the match and the best reconstructed block

Where w(block) calculates the log prob sum of all (1-positional confidences) in a block
Where $m(X)$ is the mass of X

Corresponding part of the overall formula

1. X = peptide block
2. Z = protein block
3. score ← -cache3a(X) // $-\log(P(m(x)))$
4. score ← score + cache3b(m(X),Z) // $\max_{\mathrm{m(y)}}(\log P(y) + al(y,z))$
5. score ← score + w(Z) // $\log w(x)$

Therefore, Case 3 becomes a simple case of multiple $O(1)$ lookups and which can be combined with Case 2 to give us the final algorithm that we will use for search and reconstruction.

Algorithm: Overlap Match

Input: One *de novo* X peptide sequence and a suspected protein peptide sequence Z match
Output: Score for the match, matrix allowing for one-the-fly traceback

```
1.  X = de novo peptide sequence
2.  Z = potential protein sequence match
3.  initialize matrix M of size M(|X|, |Z|) where M(0,0) = 0, M(i,0) = 0 where 1 ≤ i ≤
    |X|, M(0, j) = 0 where 1 ≤ j ≤ |Z|
4.  FOR(i from 1 to |X|) DO                                          // ~O(|X||Z|)
5.      FOR(j from 1 to |Z|) DO                                      // ~O(|Z|)
6.          maxScore ← -∞
7.          FOR(i' from i-3 to i) DO                                 // ~O(1)
8.              FOR(j' from j-3 to j) DO
9.                  compute score1 using lookup table for Case 2
10.                 compute score2 by formula for Case 3
11.                 score = M[i',j'] + max(score1,score2)
12.                 maxScore ← max(maxScore,score)
13.         M[i,j] ← maxScore
```

The resulting algorithm is $O(|X||Z|)$ and can be compared with the previous naive implementation of Spider that had a runtime of $O(|Z|^4 + |Z|^2 \times m \times |\sum|)$.

4.2 Software Implementation

We will now describe how the theoretical polynomial search was implemented as an actual program. First, it is possible to pre-calculate certain lookup tables that contain elements of our alignment score in order to improve performance. The first pre-calculated matrix is a lookup table that allows us to look up the alignment score between y and z using a BLOSUM90 matrix. This matrix matches all 1-mers, 2-mers, and 3-mers with all other possible n-mers and remains fixed regardless of PTM selection since the BLOSUM matrix does not contain information on the mutation likelihood of PTMs. The second pre-calculated matrix caches the calculated value of $\log(p(m(x)))$ for all n-mers. In other words, we cache the log probability that that we chose a combination of amino acids with the same mass as a particular n-mer. This allows us to quickly calculate case 2 of our scoring function. The last pre-calculated matrix stores $\max_{m(y)} (\log P(y) + al(y, z))$ for all pairs of mass value y and n-mer z. In other words, this maps between a potential mass segment[13] and database block to the best

[13] Note that when calculating potential mass values, we use the simplifying trick of calculating the mass of all amino acid blocks by summing the integer mass of the amino acids (or PTMs) involved. This allows us to quickly group together blocks with roughly the same mass

possible real block and its score. While the first matrix is insensitive to PTMs, the latter two are highly dependent in terms of size and processing time on the number of possible PTMs. These three pre-calculated matrices are stored on disk to save on calculation time.

When performing the actual search, as with the previous iteration of Spider, a three amino-acid seed hash structure based on the query sequence was created. Next, we go through the protein database and identify potential hits based on the hash hits.

For performance reasons, these hash hits are examined via a number of heuristics which have to be passed before we assess hits using the new model. These include using a minimum threshold for relevant candidates from the previous non-gapped homology match mode (set empirically at 27), for longer peptides (with length greater than 12) two seed matches are required, and a small LRU (Least Recently Used) cache that stores *de novo* sequence and database matches (since certain combinations will show up repeatedly, typically in a small m/z range).

After passing these heuristics, a segment of the protein corresponding to the hash hit is matched to the *de novo* sequence by mass and passed off to the overlap matching algorithm which takes advantage of the pre-calculated matrices. We align a *de novo* sequence and a peptide sequence using a combination of ideas from the pre-calculated matrices and what is called an overlap alignment [28] as shown in Figure 18. This can also be called a fit-alignment since the entirety of one sequence, the *de* novo sequence, is expected to be contained within the larger protein sequence.

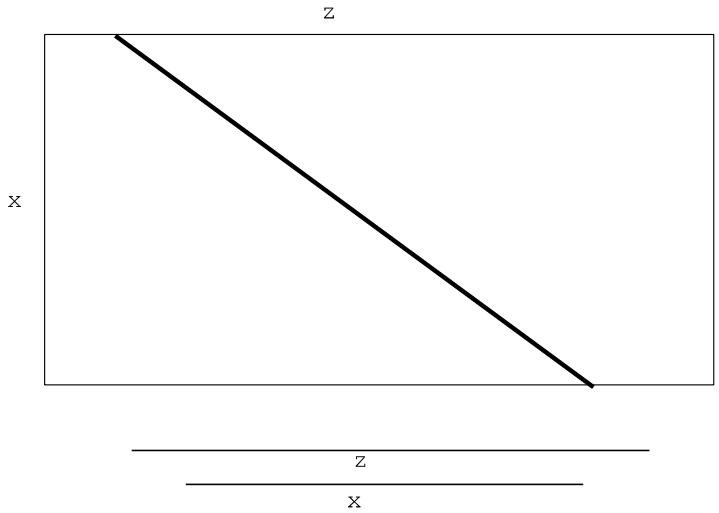

Figure 18: Example of overlap alignment from Durbin et al. [28]

As with the other types of pair-wise alignment, the algorithm works using a dynamic programming approach. Reconstruction of the "real" sequence is done by iterating through the matrices using a trace-back approach.

4.2.1 Software Details

The classes that are involved with Spider in the current implementation are SpiderWrapper, BlockMatch, HeuristicHomology, Reconstructor, SpiderAlign, OnTheFlyReconstructor, RealHomologyMatch, OverlapMatch, and PreCalculator.

The following classes are mainly relevant for the new search mode and reconstruction

- RealHomologyMatch: This class implements a full naive approach to reconstruction which ends up being relatively slow since it is an $O(|Z|^4 + |Z|^2 \times m \times |\sum|)$ algorithm. Historically, this was used as both a baseline (for example, to demonstrate the utility of reconstruction in 4.3.1) for comparing the performance of SPIDER II for search and for reconstruction. In the final version of Spider II, the following classes replace all of its functions.
- PreCalculator: This class pre-calculates the three lookup tables required to implement the new polynomial homology match algorithm. Specifically, it first creates a BLOSUM lookup matrix for 1-mers, 2-mers, and 3-mers. It then creates a lookup table that stores the log probability that we choose a combination of amino acids with the same mass as an input n-mer. Lastly, there is a third lookup table that allows us to look up the maximum possible score in the case that we chose the wrong amino acid for a mass segment.
- OverlapMatch: Implements the overlap match at the core of the algorithm. Reconstruction of the "real" sequence is done by iterating through the matrices using a trace-back approach.
- BlockMatch: Implements the actual search algorithm by tying together the OverlapMatch algorithm with heuristics for finding appropriate seeds for further investigation.
- Reconstructor: Support class which manages the trace-back feature of OverlapMatch allowing for recording reconstructed sequences

- SpiderWrapper handles the integration of Spider with the rest of the PEAKS Studio software, HeuristicHomology handles part of the heuristics for evaluating potential seeds, and SpiderAlign stores potential results

One measure we will be using to evaluate our results is RSD (Relative Sequence Distance) [36]. RSD allows us to evaluate the distance between a *de novo* sequence and the true sequence with 0 indicating a perfect match and 1 indicating a sequence which is completely different.

4.3 Experimental Procedure

4.3.1 Preliminary Results from Prior ASMS Posters

The author demonstrated a preliminary version of the new scoring function as an ASMS (American Society of Mass Spectrometry) poster, "Novel Scoring Function Improves Homology Searches Using MS/MS *de novo* Sequencing Results [37]." This poster demonstrated the ability of a preliminary version of our new scoring system to discriminate between correct and incorrect matches. We were also able to demonstrate that reconstruction substantially improves the average score by selecting the correct real amino acid sequence without significantly changing the runtime of the search. This implementation was relatively quick since reconstruction was only used to reconstruct the match for each final match.

The experiments were conducted in the following manner. First, a dataset was created from a sample of bovine serum albumin, a sample of 18 purified proteins, and a sample of *S. cerevisiae*. Spider searches were done against the human genome and *S. pombe* respectively in order to determine the effectiveness of the search and the reconstructed sequence when compared with matches of high confidence when searching with a traditional database search algorithm on the "correct" database.

As a visual, we can consult Figure 19 which depicts this workflow in general. Similar workflows will be used for all the evaluations of Spider II. However, we will use a number different pairs of real samples and homologous samples in this thesis including respectively *S. cerevisiae* (yeast)-*S. pombe* (fission yeast) here and *B. taurus* (cow)-*H. sapiens* (human) for Spider II. Champs in Chapter 5 will use a very similar workflow with *B. taurus* (cow)-*O. aries* (sheep) and *G. gallus* (chicken)-*C. japonica* (Japanese quail) as pairs.

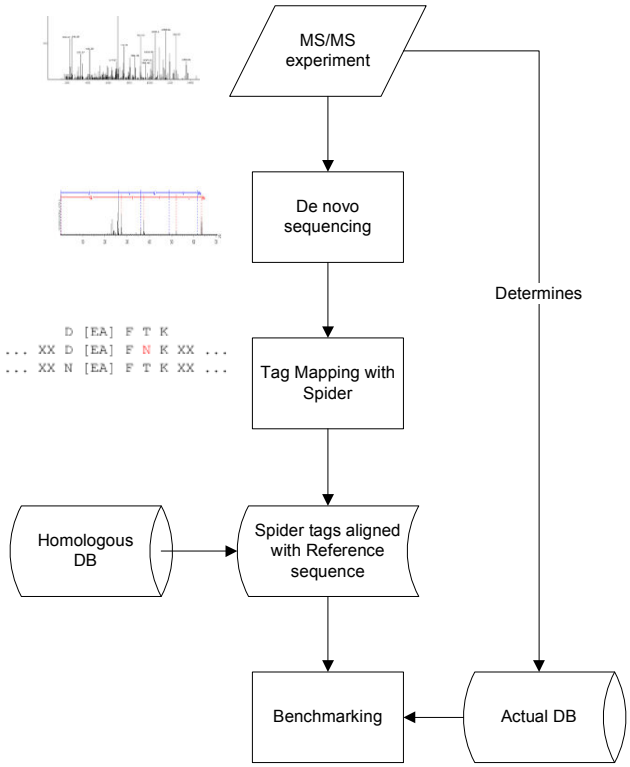

Figure 19: Spider II experimental workflow

At that time, the original Spider modes were used for the search and a precursor to the new score was applied as a post-search step to make returned matches from each of the Spider modes comparable. We found that the new score was very successful at distinguishing between incorrect and correct matches with approximately 30% of matches being correct between a score of 20 to 25, about 60% of the time between 25 and 30, and 80% above that. The result was a superior ROC (Receiver Operator Characteristic) curve which demonstrated a superior ability to choose a threshold that could balance between true positives and false positives. One example is the improvement shown in Figure 20 when going from non-gapped homology (ngap) to non-gapped homology rescored with the new scoring function (ngap-r).

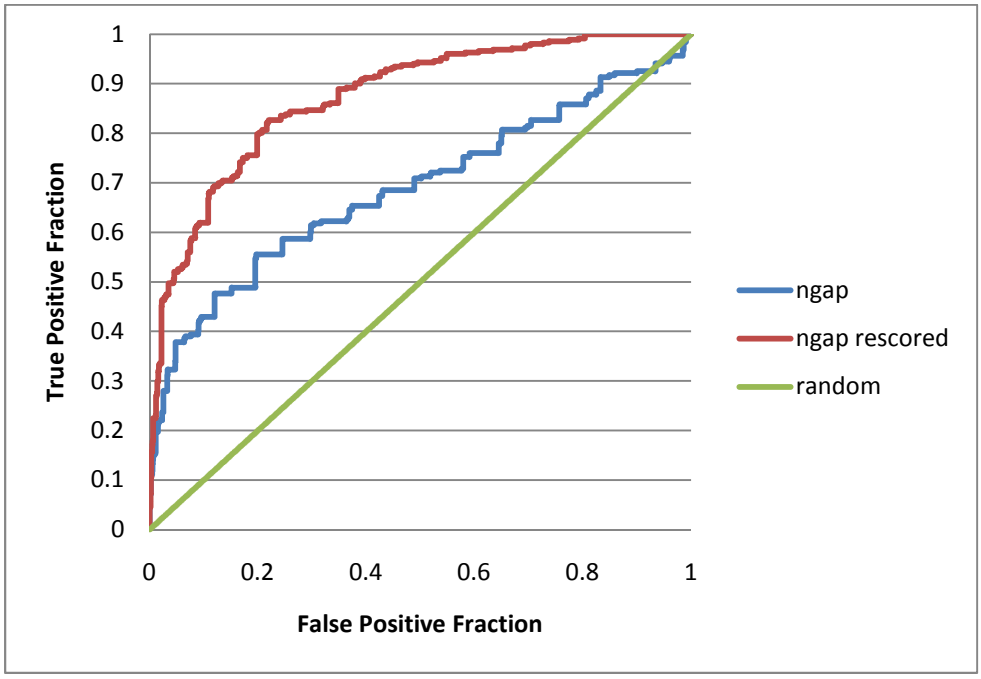

Figure 20: Non-gapped homology search rescored

4.3.2 Heuristics for Polynomial Search

Unfortunately, while the new search alignment algorithm is faster than previous heuristic methods for determining the score of a possible match, it is still not possible to use it to analyze every possible peptide-sequence match in the database. Therefore, we have developed a number of algorithms to remove matches that should be of less interest based on quicker and simpler analyses of their attributes. We will present a few example runs while varying the parameters in these heuristics to demonstrate how sensitive the scoring scheme and performance is to these changes.

It is also important to note the importance of separating the pre-calculation of the various matrices from the runtime. Performance benchmarking indicated that the newer strategy of pre-calculating the three pre-calculated matrices in 4.2 resulted in a significant runtime reduction to only 0.18% to 1.69% of the older on-the-fly reconstruction technique used in 4.3.1. This further works out to an average of 0.54% of the previous runtime when not including the pre-calculation time and 6.55% when including it. In practice, the first runtime is more common since the pre-calculated data can be cached on disk, is retrieved with trivial runtime, and only needs to be calculated once for each specific selection of PTMs.

The first dataset used is from a LTQ-Orbitrap instrument running against BSA (Bovine Serum Albumin) with multiple enzymes used. A subset of matches identified with high confidence (> 85%) in PEAKS Protein ID was searched against a homologous database in order to simulate a typical run of Spider when the real database is not available. Further details on this dataset are available in 5.3.

In Table 2, we demonstrate the changes in runtime as we vary the score threshold that is used to filter out possible seeds when running Spider in one of its simpler modes as a first pass. We can see that there is no significant change in the number of correct amino acids as we increase the score threshold used until we reach 45. This seems to indicate that this threshold is relatively robust while having a very significant effect on the runtime.

In Table 3, a LRU (Least Recently Used) cache is introduced for recently encountered *de novo* sequence/database sequence pairs. This cache should not and does not have any effect on the results but does lead to a substantial performance boost.

Table 4 displays the effect of varying the threshold at which we require two seeds of three amino acids when considering "long" matches. Experiments were performed using between six amino acids and fourteen amino acids. This showed that there is no real effect on the accuracy of results while greatly influencing the runtime.

In each table, the most important measurement is the last column which indicates the number of possible seed matches that make it past the particular heuristic we are currently examining. Other columns that may require explanation: "threshold" indicates the number of seed matches that make it past the heuristic homology threshold and "pre-cache" indicates the number of seed matches that make it to the LRU cache.

In summary, it can be concluded that that the various heuristics that we use are relatively robust to changes in the arbitrary parameters, which indicates performance can be tuned a great deal without sacrificing too much sensitivity.

# Correct AAs[14]	Exact Matches[15]	Number of Matches[16]	Heuristic Threshold[17]	Time (s)	# Seeds Passing Threshold[18]	% Seeds Passing Threshold
3669	129	401	21	625	30,131,162	28.71%
3666	129	401	27	150	5,914,602	5.64%
3666	129	401	30	90	2,368,712	2.26%
3656	130	400	35	54	529,996	0.51%
2563	86	261	45	46	40,524	0.04%
1061	52	91	55	48	18,981	0.02%

Table 2: Heuristic threshold comparison

# Correct AAs	Exact Matches	Number of Matches	Cache size	Time (s)	# Seeds After cache[19]	% Seeds Passing Cache
3666	129	401	10 cache	150	1,239,761	72.96%
3666	129	401	100 cache	144	1,152,362	67.82%
3666	129	401	1000 cache	136	1,014,549	59.71%
3666	129	401	5000 cache	119	797,881	46.96%
3666	129	401	10000 cache	111	664,741	39.12%

Table 3: Cache size comparison

[14] Correct AAs refers to the number of amino acids correctly sequenced
[15] Exact matches refers to the number of matches that were totally correctly sequenced
[16] Number of matches refers to the total number of matches returned
[17] Heuristic Threshold refers to the threshold chosen to restrict possible seeds before the main polynomial search
[18] The last two columns refer to the number of seeds that past the threshold both as a # and as a percentage
[19] Here, the last two columns refer to the number of seeds that make it past the cache.

# Correct AAs	Exact Matches	Number of Matches	Two hit threshold[20]	Time (s)	# Seeds after two-hit threshold[21]	% Seeds Passing two-hit threshold
3655	125	401	6	59	373,834	6.33%
3655	125	401	8	60	393,479	6.66%
3657	125	401	10	63	594,818	10.07%
3666	129	401	12	116	1,699,392	28.76%
3797	127	415	14	294	4,871,340	82.44%

Table 4: Two-hit threshold comparison

[20] Two hit threshold refers to the length of peptide that requires two seeds
[21] Here, the last two columns refer to the number of seeds that make it past the two-hit threshold

4.4 Results

4.4.1 Spider II vs. Spider

Our main experiments will compare the results of Spider (comprising the segment, non-gapped and gapped match search modes) and the results of Spider II (using the new homology search mode). We will use *de novo* sequences from PEAKS Studio 4.5 in both cases in order to hold the *de novo* sequences constant[22].

Due to changes in the software over the years, a slightly convoluted testing procedure was implemented in order to determine the differences between the old Spider search modes and the new Spider II search modes due to changes in file formats and output. Thus, the procedure will be similar to the following:

1. Run *de novo* in PEAKS Studio 4.5
2. Run Spider in PEAKS Studio 4.5 and store the results to anz file format
3. Import *de novo* results and run Spider II in a modified version of PEAKS Studio 5.2
4. Export all results to pepXML
5. An automated testing framework was created and modified to handle Spider results specially, by parsing reconstructed sequences in addition to the reported database match. This will give us overall performance characteristics including number of correct amino acids, correct matches, etc.
6. Output both sets of results to Excel using a modified version of Peaks (Modified to allow for output in old file formats) including homolog and reconstructed results. This will allow us to examine individual results and differences for evaluation

For comparison purposes, it is also possible to run DirecTag and then TagRecon for comparison purposes. This software is run with charge state from the data (UseChargeStateFromMS = true) and the appropriate mass tolerance (PrecursorMzTolerance = x) to make the results comparable to Spider. In the two cases, the testing framework will evaluate the top scoring and first result respectively for

[22] If we did not hold *de novo* sequences constant, we would be comparing the results of Spider on *de novo* sequences from different versions of Spider which would be unfair due to improvements in the *de novo* sequencing algorithms

each result. The comparison also includes Mascot error-tolerant search with the equivalent parameters.

MS-BLAST was also considered for this comparison. However, it was only available as a web interface online and it also did not have the required protein databases, therefore it was excluded in the following high-throughput comparisons. However, a performance evaluation was done on a smaller data sample in the original Spider paper[3] as shown in Table 5. In this evaluation, 144 ion trap spectra were obtained from multiple proteins contained in the SwissProt database. However, the human database did contain proteins that were homologous. Therefore, the segment match mode was most effective for the SwissProt database (since it does not consider mutations) whereas the gapped homology search was most successful on the human database. In all cases except for the exact match mode, Spider outperformed MS-BLAST. This gives us an idea as to the performance difference before we start to consider the improvements made in Spider II.

	SwissProt	Human
SPIDER exact	35	13
SPIDER segment	**90**	36
SPIDER non-gapped homology	76	49
SPIDER gapped homology	78	**52**
MS-BLAST	52	24

Table 5: Comparison of SPIDER and MS-BLAST

In the following experiments, plain *de novo* results are also presented to give an idea of what a pure *de novo* result would yield as a baseline. In some cases, we will expect the correction of a "noisy" related species database to make the result worse by introducing error (coming from too many mutations). Hopefully in more cases, the database will provide the correct information to "correct" *de novo* error.

4.4.2 ABRF sPRG2006 Dataset

The first result from this procedure would be from the ABRF_FT-TRAP dataset. This dataset is part of a larger data set provided by the Proteomics Standards Research Group at the ABRF (Association of Biomolecular Facilities) [38]. The mixture contains 49 human proteins and was used as an

objective standard to evaluate around 120 research laboratories that processed the data on various instruments (and with different levels of expertise).

In the following example, the dataset was searched against a bovine database in order to simulate the normal operation of Spider when the real database is unavailable (but a related database is available). Results are benchmarked against the correct answers as determined by a high quality database search run against a human database.

		Correct AA AVG[23]	Correct AA SUM	Edit Distance AVG[24]	Edit Distance SUM	Exact Matches SUM	Incorrect AA SUM	Number of Matches SUM	RSD AVG
	De novo baseline	7.41	3325	4.53	2036	138	1698	449	0.33
H. sapiens	PEAKS DB	8.59	3857	0.96	429	329	391	381	0.09
B. taurus	PEAKS DB	3.67	1648	1.39	624	127	585	211	0.13
	Tag Recon 1.2.34	3.57	1602	7.41	3325	42	3161	409	0.61
	Mascot 2.3.01	4.72	2118	2.90	1302	119	1199	319	0.27
	Spider Segment Match	5.46	2453	6.45	2896	117	2611	449	0.51
	Spider Non-gapped Homology	6.85	3076	4.90	2200	101	1550	449	0.39
	Spider Gapped Homology	6.91	3102	4.83	2168	93	1884	449	0.38
	Spider II: Tag Match	5.68	2551	6.35	2851	125	2579	449	0.50
	Spider II: Homology Match	6.37	2861	5.15	2314	91	2053	436	0.40
	Spider II: Homology Match with Recon.[25]	7.44	3341	4.01	1801	164	1516	436	0.30

Table 6: ABRF FT-TRAP Result Comparison

[23] AVG refers to an average of the statistic in question across all reported peptides while SUM is the sum
[24] Edit distance refers to the number of steps/residues that would need to be taken to change from the result to the control sequence
[25] Short for reconstruction

Table 6 gives us a comparison between Spider I and Spider II as well as a leading database search technique.

One observation that can be made in this initial analysis is that not many of the approaches do better than the *de novo* baseline reported at the top in terms of exact match, RSD, or number of correct amino acids. This seems to indicate that the database is misleading (as we might expect due to it being from a different species). In fact, only Spider reconstruction manages to use the homologous database to manage a modest drop in RSD and a modest increase in the number of correct amino acids (and a decrease in the number of incorrect amino acids).

We calculate the number of exact matches by doing a straight string comparison between the sequence returned (whether it is a tag, *de novo,* database sequence, or reconstructed sequence) while ignoring I and L or K and Q substitutions. We calculate the number of correct amino acids by building an optimal alignment and counting the correct amino acids.

Another trend is that the number of exact matches increases as we move from tag matching to homology approaches. As one might expect, when the search space is expanded to include the possibility of mutations, the number of exact matches may drop as compared to a more conservative algorithm. One interesting counterpoint is that the Mascot error-tolerant approach returns relatively fewer matches but has better accuracy on the matches that are returned. By contrast, Spider II returns significantly more correct amino acids with both reconstruction or without while returning significantly fewer complete matches. This seems to indicate that Spider II is much less conservative, returning a match with some correct amino acids even if they are incomplete.

Indeed, when we examine the percentage of correctly identified peptide matches (sorted by score) across the dataset in Figure 24, we find that the conventional methods like tag search and database search have an advantage in this dataset. However, this advantage is much reduced in later datasets which are more difficult in terms of data quality, especially in LTQ2448 which is data from an ion trap. Note that *de novo* sequencing correctly sequences roughly 31% of matches in this dataset as opposed to 11% in LTQ2448. Another promising result is that the Spider II homology match performs significantly better than the previous homology match with a better ratio of correctly identified peptides at high scores. This indicates that we have a better measure that can separate less confident peptides from those with high confidence. This was part of the reason we chose the current scoring formula rather than the alternate scoring scheme described in 4.1.3 which was unable to make this distinction. In that scoring scheme, high scores were disproportionately given to longer peptides

rather than correct peptides due to its version of Case 2 and the term $-\log P(X)$ which was added to all correct amino acid matches.

4.4.3 BSA Champs Dataset

Another dataset that we will benchmark Spider on will be the dataset that was used for testing Champs (and is covered in detail in section 5.3). In this experiment, BSA (Bovine Serum Albumin) was digested with three different enzymes (GluC, LysC, and trypsin) in order to obtain overlapping peptides. The digested sample was then analyzed with an LTQ-Orbitrap hybrid instrument from Thermo Scientific in order to generate our data.

By using a subset of matches with greater than 85% confidence in traditional database search on the correct BSA database, we were able to separate out only those spectra (and *de novo* sequences) that correspond to real database matches. These resulting 423 spectra (out of a possible 5154) were used for a further test of Spider in its various incarnations.

Table 7 illustrates some further trends between the various approaches. On the face of it, this dataset displays some different trends for all the various algorithms. Despite the equivalent size of the dataset, the number of exact matches for all approaches is as small as or smaller than in the previous dataset. Since the RSD numbers are comparable, this is probably due to the presence of much longer peptides. As confirmed by both the *de novo* baseline and a PEAKS database search on the "correct" species, the average peptide is roughly 1.3 amino acids longer in this dataset when compared with the previous dataset. Again, many of the search methods return a lower number of matches than the *de novo* baseline which indicates that the database matches are returning sequences that are not very informative.

		Correct AA AVG	Correct AA SUM	Edit Distance AVG	Edit Distance SUM	Exact Matches SUM	Incorrect AA SUM	Number of Matches SUM	RSD AVG
	De novo baseline	8.72	3698	3.38	1435	94	1292	422	0.26
B. taurus	PEAKS DB	9.90	4196	0.50	211	346	199	373	0.04
H. sapiens	PEAKS DB	2.02	858	3.31	1402	18	1313	188	0.27
	Tag Recon 1.2.34	1.47	622	2.63	1114	10	1062	147	0.22
	Mascot 2.3.01	2.71	1145	6.60	2792	11	2699	321	0.52
	Spider Segment Match	4.57	1938	8.08	3425	21	3189	421	0.61
	Spider Non-gapped Homology	7.09	3007	4.98	2112	18	1568	421	0.39
	Spider Gapped Homology	7.02	2975	5.04	2137	18	1955	417	0.39
	Spider II: Tag Match	4.27	1811	8.22	3486	21	3375	421	0.64
	Spider II: Homology Match	6.85	2903	4.71	1997	17	1820	401	0.37
	Spider II: Homology Match with Recon.	8.65	3666	2.87	1216	129	1054	401	0.21

Table 7: BSA85 Comparison

However, in this dataset we again see a large bump in the number of successfully reconstructed sequences as measured by number of correct amino acids, exact matches, and average RSD. In fact, it should be emphasized that reconstruction manages to return the highest number of exact matches while maintaining the lowest RSD and a very low number of incorrect amino acids. In other words, despite returning more matches than a number of other approaches, Spider II still manages to return a lot of useful information.

The second row of the table shows the theoretical maximum when a database search algorithm is run against the "correct" database. This is then compared to all the other results which are run on the "incorrect" homologous human database. This unfair comparison effectively gives an upper limit on the theoretical performance of our homology search algorithms. No matter how good a homology search algorithm is, it should not outperform a standard search on the real database. Spider II reconstruction does very well on this baseline attaining 87% as many correct amino acids.

In Figure 25, we also see another success of this approach. Reconstruction retains a superior % of correct peptides over and above all other search engines when results are sorted by score. Additionally, the search mode of Spider II again does a better job than the previous homology match in assigning high scores to correct matches.

4.4.4 LTQ2448

Another dataset that we will benchmark Spider on is the LTQ2448 dataset from the MSNovo paper [39]. It contains protein hits from the MSDB *Saccharomyces cerevisiae* database. We will perform a search against Schizosaccharomyces pombe. LTQ2448 appears to be a more challenging dataset since the divergence happens at a subphylum classification as opposed to the divergence on the order classification as in the previous datasets.

		Correct AA AVG	Correct AA SUM	Edit Distance AVG	Edit Distance SUM	Exact Matches SUM	Incorrect AA SUM	Number of Matches SUM	RSD AVG
S. cerevisiae	*De novo* baseline	6.87	16812	6.34	15526	269	13048	2448	0.44
	PEAKS DB	9.88	24188	0.25	621	1875	556	1955	0.02
S. pombe	PEAKS DB	0.67	1629	0.30	743	103	639	230	0.03
	Tag Recon 1.2.34	1.58	3880	3.91	9582	54	8690	1040	0.29
	Mascot 2.3.01	2.70	6602	7.00	17130	96	15900	1954	0.56
	Spider Segment Match	3.85	9419	9.41	23038	89	20643	2438	0.67
	Spider Non-gapped Homology	5.23	12793	7.73	18925	82	13166	2448	0.56
	Spider Gapped Homology	5.45	13338	7.67	18768	76	16570	2445	0.55
	Spider II: Tag Match	3.78	9260	9.51	23274	87	20909	2438	0.68
	Spider II: Homology Match	5.29	12949	8.01	19614	75	17359	2422	0.56
	Spider II: Homology Match with Recon.	7.54	18448	5.77	14124	430	11604	2422	0.38

Table 8: LTQ2448 Result Comparison

First, looking at Table 8, we also see that reconstruction clearly excels in terms of the number of correct amino acids and RSD. Spider II manages to return more results while maintaining a low RSD and a high number of correct amino acids while limiting the number of incorrect amino acids. In particular, 76% of correct amino acids can be found when comparing to the artificial best-case of searching on the correct database while the RSD is significantly lower than the *de novo* baseline indicating that the homologous database is very helpful in this dataset. In Figure 26, we see the acceptance curves are somewhat in the middle of the two previous datasets. While tag searching and database search has an initial advantage at the highest scores, Spider manages to overtake the other approaches with roughly 10% of peptides accepted. It also levels off with roughly 30% of peptides successfully reconstructed as opposed to 10% for other approaches.

Looking at all the datasets at once, we see that reconstruction adds valuable information to the *de novo* sequencing with few drawbacks in terms of the results. Many more amino acids can be correctly identified and incorrect amino acids ruled out. Additionally, the score gives us a good indication of how correct the reconstructed sequence is in each case. Unfortunately, there does not seem to be a significant change in the performance of the actual homology match against the old homology match in terms of absolute results. While the new algorithm is better at separating unlikely matches from likely matches, the number of correct amino acids, exact matches, and the average RSD is very similar between the two approaches. It is possible that by optimizing for the reconstructed sequence, we were unable to significantly improve the quality of the matches.

It is also possible to consider how much overlap there is between the various groups of matches. Grouping the matches returned spectrum-by-spectrum between Mascot, Peaks, and Spider, we can examine a series of Venn diagrams that visualize which correct matches are returned by which search engine. Figure 23 demonstrates a high level of overlap between Mascot (even in error-tolerant mode) and Peaks, both database search algorithms. On the other hand, we see that Spider has a relatively minor overlap with the other search engines and contributes a large number of unique matches. This seems to indicate that Spider II results may complement other search methods by providing a lot of additional information. This observation is repeated in Figure 21 and Figure 22.

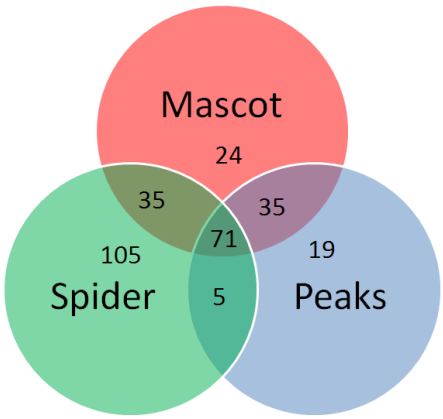

Figure 21: ABRF FT-TRAP Venn diagram (RSD ≤ 0.2)

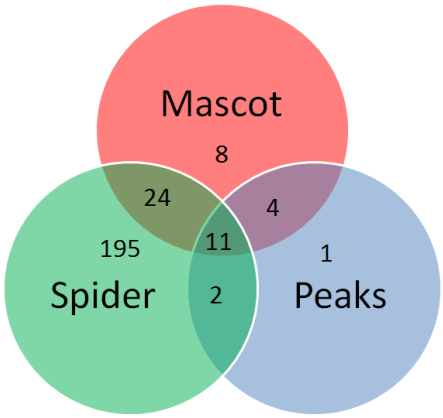

Figure 22: BSA85 Venn diagram (RSD ≤ 0.2)

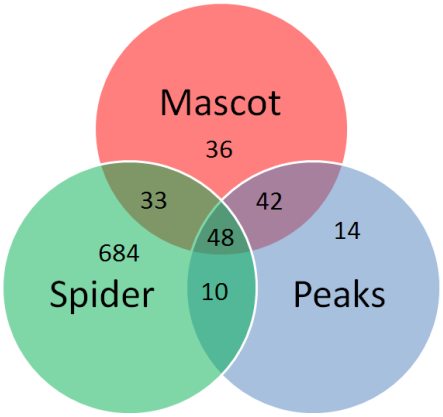

Figure 23: LTQ2448 Venn diagram (RSD ≤ 0.2)

The following charts have been analysed one-by-one as we discussed each dataset. However, we also wish to make a number of observations looking at all the charts at once. The most obvious observation is that Spider reconstruction clearly has an advantage at all but the lowest number of accepted peptides. At a very low score threshold, the Protein ID and Mascot algorithms have an advantage possibly because they are less likely to force an interpretation for a noisy spectrum. (TagRecon was not included since it does not return a comparable score) But it should be noted that Spider II retains a curve similar to error-tolerant Mascot while returning many more matches. We can also see that the Spider II search is highly competitive with the older version of Spider in both datasets, outperforming all prior search modes.

One interesting note is that the data quality score for each dataset parallels the effectiveness of Spider. Data quality can be calculated by a number of factors such as the noise to signal ratio, the number of peaks, the longest continuous sequence tag available via a simple computation that matches single residues to mass differences, and the number of sequence tags that can be assigned to a spectrum [40]. Using this measure, Spider works best relative to simpler database searching options on the dataset with the lowest data quality (BSA85) and worst on the dataset with the highest data quality (FT-TRAP). This makes sense because you would expect that there will always be some peptides that are not mutated and are relatively easy to match with data of sufficient quality while Spider and *de novo* should outperform on data that database search finds harder to match.

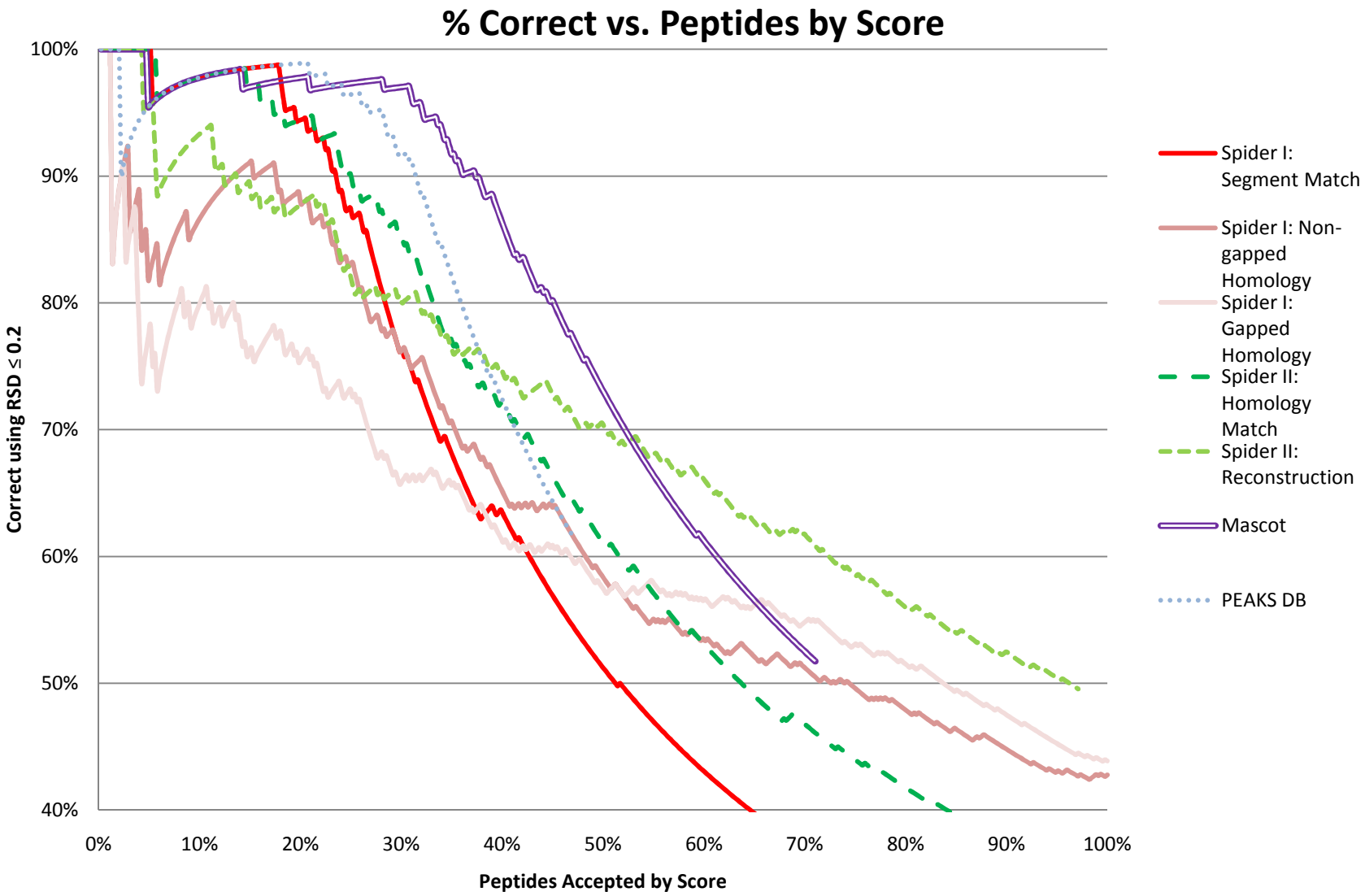

Figure 24: ABRF FT-TRAP - % Correct vs. Accepted Peptides

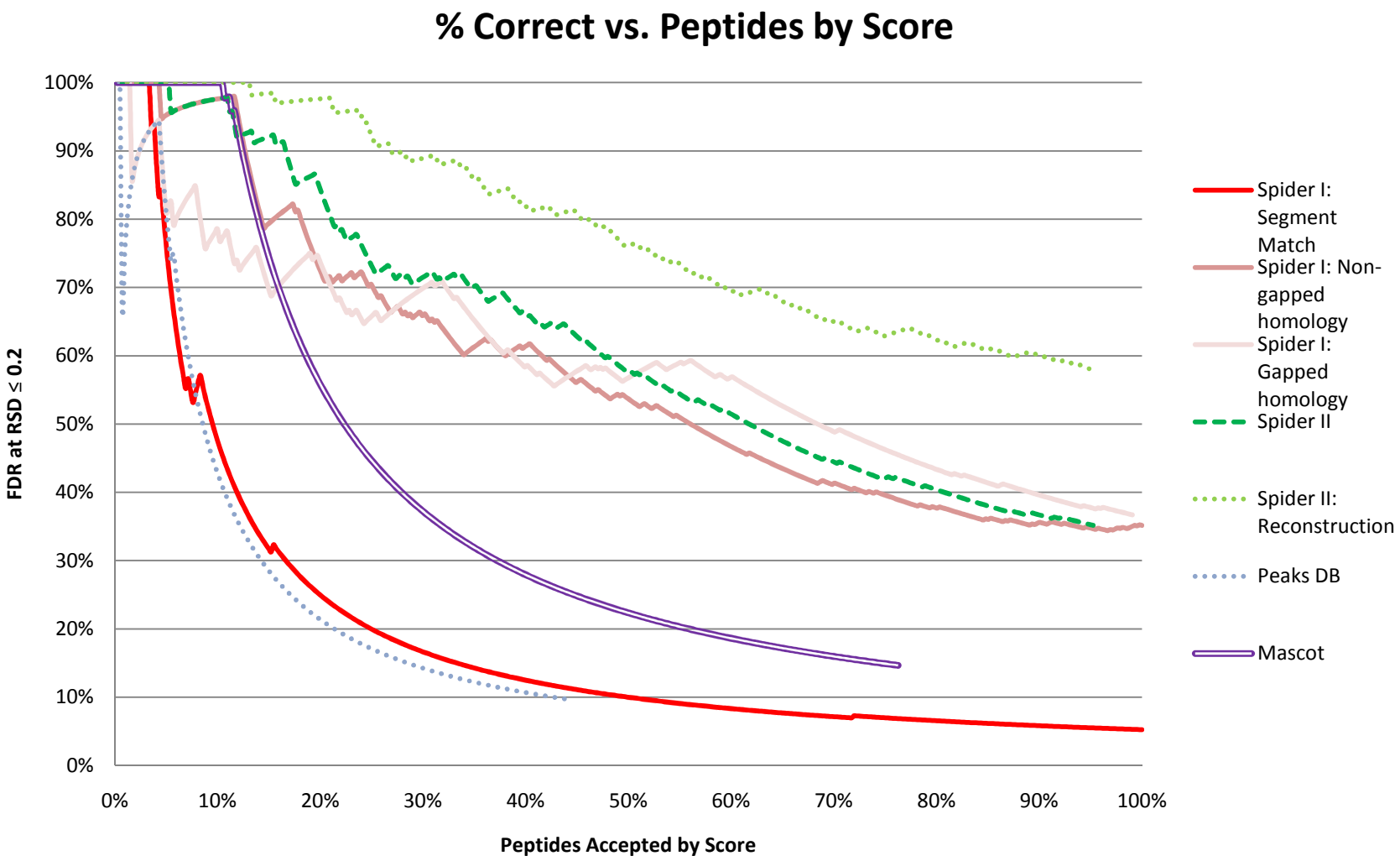

Figure 25: BSA85 - % Correct vs. Accepted Peptides

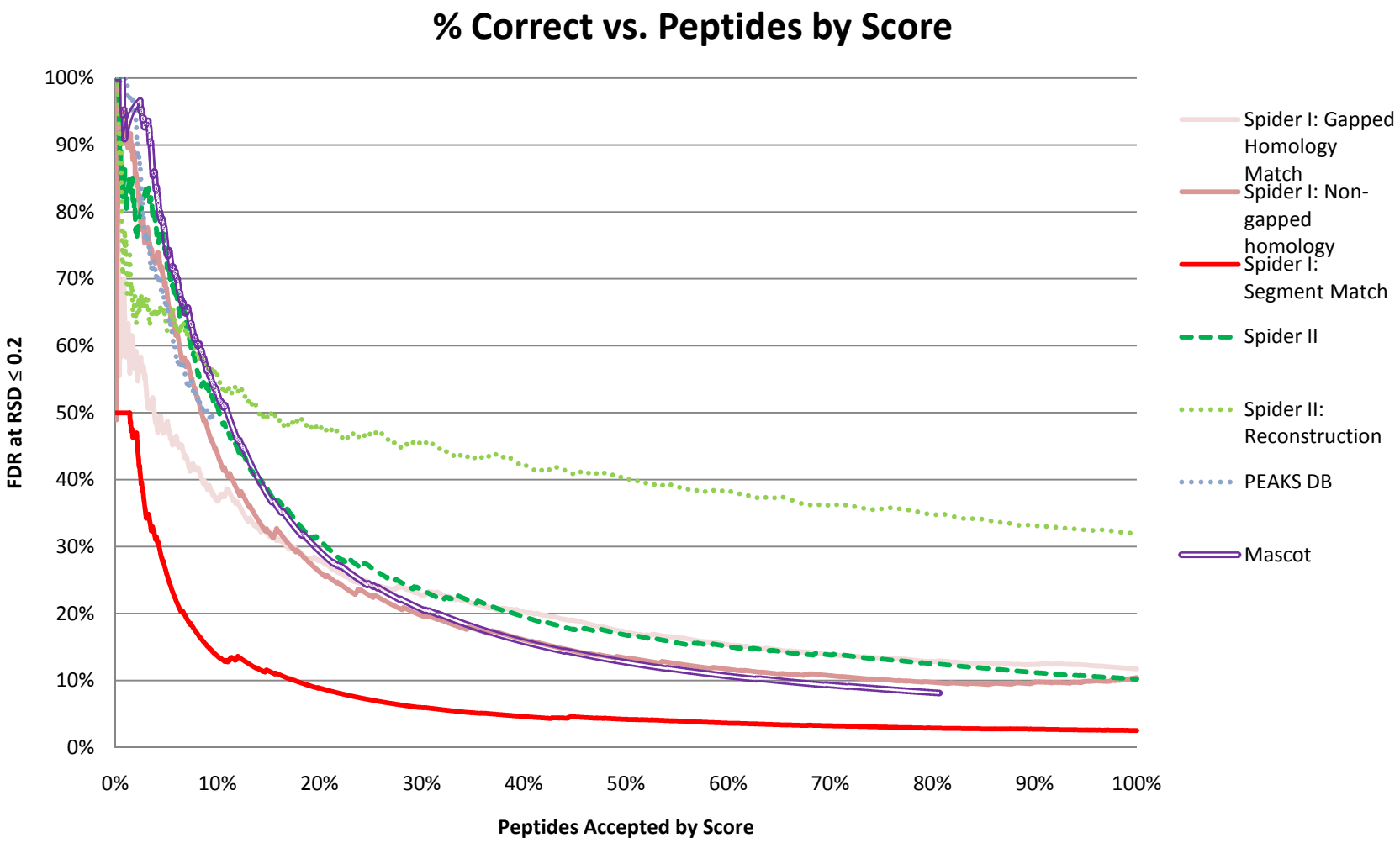

Figure 26: LTQ2448 - % Correct vs. Accepted Peptides

Finally, we can also average the counts of exact matches and correct amino acids from all three datasets. Then we will display these as a percentage of the results from the best case, running database search on the "correct" database in Table 9. Again, we see that there is a drop-off in the number of exact matches going from database search to most homology searches while the number of correct amino acids increases. The most significant result is that reconstruction manages to return more than double the number of exact matches (to 37%) and gives a significant boost in the number of correct amino acids when compared to the previous approaches.

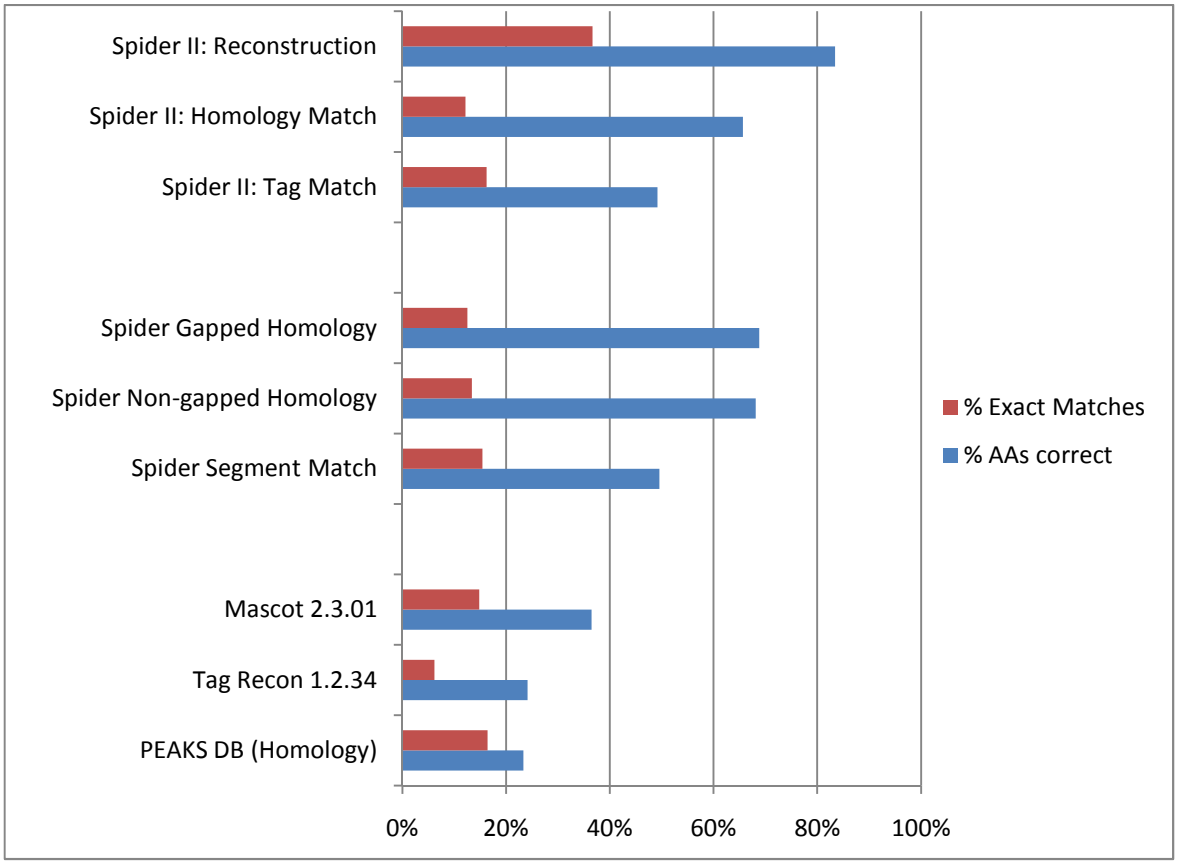

Table 9: Exact matches and AAs correct as % of best case

Chapter 5
CHAMPS

5.1 Theory

We covered the need for protein sequencing technology in section 3.2.5. In short, while a homologous database can be used to identify the function of proteins, simply knowing the function[26] of a protein is not as complete as protein sequencing which can characterize the possibility of mutations between species and between individuals as well as the possibility of characterizing PTMs.

Our approach to the problem takes advantage of Spider reconstruction and homology search via the Champs application. Its development was inspired by a Chinese saying, "兼聽則明，偏聽則暗" which roughly translates to "listen to both sides to find out the truth." This is the philosophy behind the development of Spider reconstruction which relies upon both an inaccurate *de novo* sequence and an inaccurate database sequence to reconstruct a suspected real sequence. It is also the philosophy that will be expanded in order to cover a whole protein.

The approach in Champs is an automated protein sequencing approach (some previous approaches reviewed in 3.2.5 were manually assembled) using both a homologous database and tandem mass spectra. This chapter is devoted to a broad analysis of Champs while putting it in context with the procedure of reconstruction. The algorithm and some alternate results have been previously published in a co-authored paper entitled "Automated protein (re)sequencing with MS/MS and a homologous database yields almost full coverage and accuracy."[41]

In terms of preparation on the biological side, we will require a protein that is cut with multiple enzymes. This gives us multiple overlapping peptides which increases coverage and also allows for the use of peptide overlaps to assemble the final peptides together. For example, in Figure 27 we can see peptides ending at position 212 with lysine (K) indicating that they are cut with trypsin overlapping with peptides ending at position 208 with glutamic acid (E) indicating that they are cut with GluC. Additional overlapping peptides are seen ending at positions 209 (trypsin) and 206 (GluC).

[26] More specifically, the probable function of a protein as suspected due to the function of a known homologous protein

Figure 27: Overlapping peptides

On the computational side, our approach uses a selected reference protein, performs *de novo* sequencing on all spectra, selects high-scoring sequences for tag mapping via Spider and then assembles high-scoring alignments onto the reference protein. An overview of the algorithmic steps is as follows and visualized in Figure 28:

First, *de novo* peptide sequencing is run on the entire dataset and the results are filtered for the highest scoring peptide in each spectrum. These sequences are kept if they pass certain filters which will be further discussed later.

Second, the identification of the reference protein is relatively straightforward. Either a database search or a Spider homology search can be used since the reference protein and the real protein will likely share at least some identical peptides.

Third, Spider is used to match the *de novo* sequences with the reference protein while taking into account mass tag errors and likely mutations. As discussed earlier, Spider creates reconstructed sequences which are essential for Champs to sequence proteins. However, Spider also contributes potential alignments between these sequences and the reference protein. Champs refers to these potential alignments as *de novo* tag mapping.

The last step is to use dynamic programming to process the alignments column by column in order to output a predicted protein based on the Spider tags and the reference protein. This step is also

referred to as Spider tag assembly. It is worth highlighting that Champs assembles interpreted spectra in the form of reconstructed sequences rather than assembling spectra into consensus spectra as in some previous approaches.

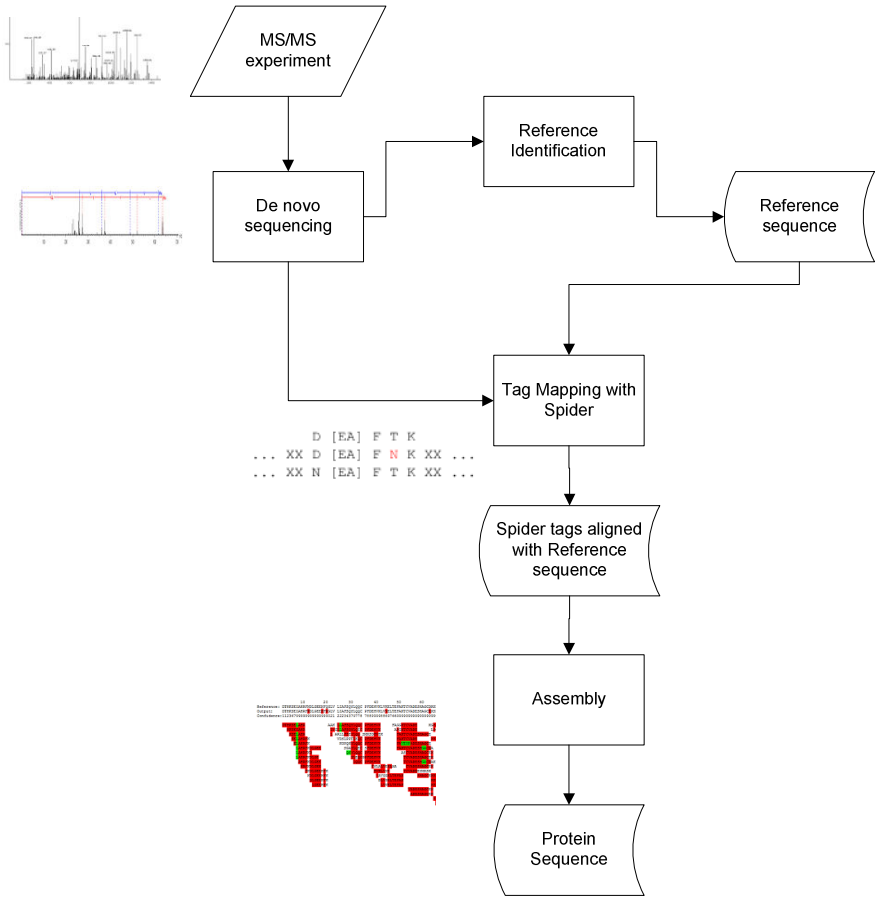

Figure 28: Champs workflow

5.1.1 *De novo* Tag Mapping

For *de novo* sequencing, we will continue to use PEAKS auto *de novo*. However, in order to map these sequences to the reference protein, we will need to consider two factors, *de novo* sequencing errors and mutations in the target sequence as compared to the reference sequence. These concerns are taken care of by Spider which maps the *de novo* tags while taking these factors into account. Additionally, with Spider II we get reconstructed sequences which can be aligned with the reference sequence.

Past this point, to avoid confusion, we will refer to reconstructed sequences as "Spider tags" for simplicity. Several filters are used to remove Spider tags that are either too unreliable or that are redundant. First, tags do not pass the following conditions, a score ≥ 10, with a length ≥ 5, and a score ≥ 0.9 times its length, are removed. Another filter is that Spider tags that are strict substrings of other tags are removed if they map to different locations. Finally, if two different tags map to the same peptide in the reference protein, we keep only the higher scoring one.

5.1.2 Spider Tag Assembly

As in Spider, we will develop a score for a predicted sequence based on the reference and Spider tags. We will hold Y consistent from Spider as the reconstructed sequence and let Z correspond to the reference sequence provided. S_{pre} is the predicted sequence that we are trying to construct. In other words, S_{pre} is the sequenced protein that we want to compute as our final output. As a parallel to our previous work, we can think of our general problem as:

$$d(Y, Z) = \max_{S_{pre}} \left(sc(A) + \alpha \times \sum_{i=1}^{n} sc(Y_i) \right) \quad (5.1)$$

We are trying to determine a predicted sequence that maximizes a score based on the *de novo* tag mapping provided by Spider involving *n* Spider tags, a simple sequence alignment score between Z and S_{pre} designated as $sc(A)$, and a BlockScore defined on the Spider tags designated as $sc(Y_i)$. Additionally, we will add a tuning parameter denoted as α chosen depending on our confidence in either the reference sequence (which implies a lower α which favours $sc(A)$) or the Spider tags (which implies a higher α which favours the Spider tags). In practice, the algorithm appears relatively robust when choosing α. Other factors such as the score thresholds of the *de novo* and Spider tags used seem much more relevant.

The BlockScore that we see as $sc(Y_i)$ is calculated based on the alignment between the Spider tags, the reference sequence, and the predicted sequence. Spider naturally creates an alignment as previously shown in Figure 9 for sequences X(*de novo*), Y(reconstructed), and Z(homologous protein). This alignment naturally (by aligning mass tags) lends itself to a block-wise alignment along with a global predicted sequence across all Spider tags as in Figure 29.

Figure 29: Contrasting reconstruction with sequencing

The reconstructed sequence Y is used along with the reference sequence (which is the same as Z) to create the predicted sequence. As for the composition of the blocks involved, we can see five blocks as displayed, the retention of T in block 4 as identified via reconstruction and the possibility of other overlapping Spider tags causing a different amino acid at block 1 to be predicted instead of either the reference or Spider tag.

This block-wise alignment is used to create $sc(Y_i)$ which is defined as a fairly simple score below:

BlockScore:

Input: A block B_j which has as its components, $S_{pre}|B_j$, $Z|B_j$, and $Yj|B_j$ as the three parts of the alignment
Output: A block score

Where $m(B_j)$ gives the mass of a block

1. If $(|B_j| > 3)$ then $f(B_j) = 0$
2. ELSE IF $S_{pre}|B_j = Yj|B_j \neq$ empty string then $f(B_j) = |B_j|$
3. ELSE IF $m(S_{pre}|B_j) \approx m(Yj|B_j)$ then $f(B_j) = 1$
4. ELSE $f(B_j) = 0$

This score has several features. The length constraint matches the reasoning for that given in Spider II. As the mass of a block increases, the number of other random blocks with the same mass grows at an increasing rate. Past a length of three amino acids, the tag provides little evidence. This trend can be partially seen in Table 1. By restricting our calculations to a block of three amino acids, this also aids us with a definition for the recurrence relation.

Another piece that we will need is a way of looking at the problem of how to align multiple Spider tags to the reference sequence by inserting gaps. One example is presented below in Figure 30. This example shows an alignment between the reference sequence FGK and the Spider tag ATFR. In this representation, each a_j is a letter from the reference sequence while each b_j is either a letter from the Spider tag or a gap. Each Z_j can represent a gap inserted in the alignment while each Y_j represents a possible empty substring of Y.

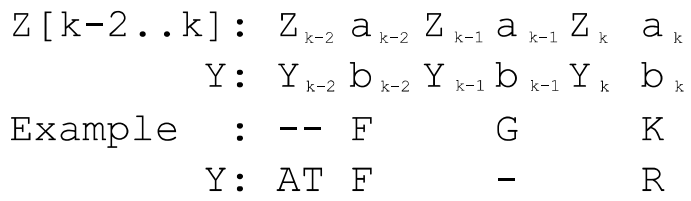

Figure 30: Sample alignment

5.1.3 Algorithm

With all these pieces in place, we can start describing the algorithm for the Champs problem. We should consider the problem as follows, given a reference sequence *Z*, *n* Spider tags *Y1, Y2,... Yn*, and the alignments of those Spider tags with a reference sequence of size *m*, we wish to construct an optimal sequence S_{pre} and an alignment *A* with *Z* so that sc(S_{pre}, A) is maximised.

As with Spider II, we can calculate this by using dynamic programming. Going back to our block concept, we will also denote B_{ij} for $1 \leq j \leq m$ and $1 \leq i \leq n$ to represent the blocks of the alignment. For simplicity, we will use B_j to denote all the blocks in the Spider tags aligned at position *j* in the reference sequence. Moving through the reference sequence by iterating through the set of blocks at each position, we will examine each set of blocks B_j. We will build up our solution by considering a prefix of S_{pre} predicted from Z[1..k] and all blocks that that end before *Z*[k], which we will denote as B_j<k. Similar to our proof of Spider II in 4.1.4, we will be able to restrict our computation at each step by keeping in mind that we only have to consider (≤ 3)-mers. This is reflected by our definition of the BlockScore. Thus, we will let E be a (≤ 3)-amino acid alignment between a portion of a potential Spider tag and the reference sequence as shown in Figure 31. As in the previous section *Z* and *Y* denote the potential gaps that may be inserted in order to establish an alignment.

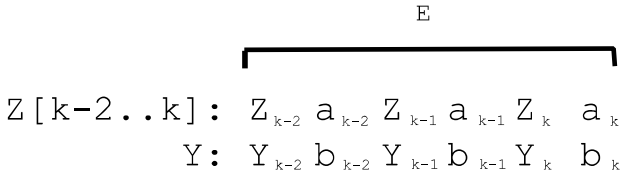

Figure 31: Illustration of E

We can now define what our algorithm should compute.

$$DP[k,E] = \max_{A' ends\ with\ E} \left(sc(A') + \alpha \times \sum_{B_j<k} f(B_j) \right) \quad (5.2)$$

Given this definition, the final solution for the general problem defined by (5.1) will be equal to $\max_E DP[m,E]$. In other words, the maximum score will end with all possible E and at the end of the reference sequence. This gives us both the maximum score and an optimal alignment if we either store our computation at each step or use a traceback procedure. We will now create a recurrence relation that can compute $DP[k,E]$ efficiently. In order to do this, we will consider the last few columns of the alignment.

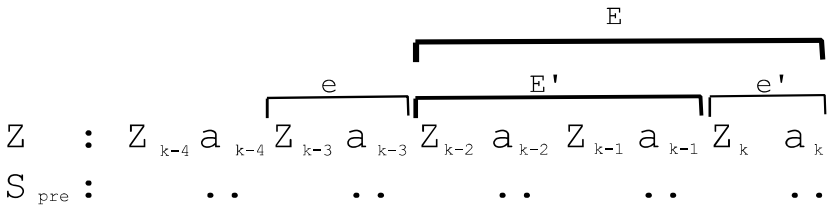

Figure 32: Alignments in recurrence relation

Thus, we can see that E can be broken up as E' and e' which are preceded by e in the optimal solution. Thus, we can assemble *DP[k,E]* from *DP[k-1,eE']* given the block score of B_j where k-1≤ B_j≤ k and the alignment score for the alignment of e'. This gives us the following recurrence relation.

$$DP[k,E] = \max_e \left(DP[k-1, eE'] + sc(e') + \sum_{k-1<B_j<k} f(B_j) \right) \quad (5.3)$$

We can lookup *DP[k-1,eE']* and *sc(e')* can be calculated using BLOSUM. The value $f(B_j)$ can be calculated from eE since due to our definition of BlockScore, we do not need to look back further than three letters since a block greater than length 3 has a BlockScore of 0.

Finally, we can note that when we choose an optimal e, we can introduce as a parameter, at most c insertions into Z_{k-3}. As each insertion can be one of 20 amino acids, we get a possible $(20+20^2+...+20^c)\times 21$ choices where the last column corresponding to a_{k-3} can be one of 20 amino acids or a gap. This gives $O(21^{c+1})$ for the number of possible E while a similar analysis gives a bound of $O(21^{3c+3})$ for possible alignments of E.

Therefore, since we control c, we can establish a solution of the form $O(|\sum| + 1)$ or $O(1)$ for each step and a $O(m)$ linear time solution for the problem as a whole.

5.2 Software Details

Champs was implemented as a Java program, implementing the classes that calculate the assembly problem and a wrapped version of Spider that tackles the alignment and reconstruction problems. The core algorithm is implemented in seven packages, algn (handles pair alignment), assm (assembly of alignments to the reference using dynamic programming), enzy (enzyme), ion, mat (handles the BLOSUM 90 matrix used for scoring), pep (peptide), and res (amino acid residue). Two packages for utilities and the wrapper finish the implementation.

- The following classes are particularly relevant
    - ChampsFacade: This class co-ordinates input from a pepXML parser that reads *de novo* sequences, the reference protein, and manages the alignment algorithm
    - DP: Standing for dynamic programming, this class assembles chosen pair-aligned *de novo* sequences with the protein and calculates via back-tracking, an optimal sequence for the entire protein
    - RefAlign: Given a homolog segment and a Spider tag, this class gets the optimal alignment between the Spider tag and the reference tag.

5.3 Experimental Procedure

A programming interface was created for testing Champs. This interface would allow Champs to write out the reconstructed sequence and compare it with a known sequence. Champs will be tested in ways that are parallel to the testing of Spider peptide-by-peptide. We will compare the number of successfully identified amino acids versus the number of unsuccessfully identified amino acids as a measurement of accuracy. It might also be useful to track whether a missing mutation is possible to recover in a block from aligned peptides. If the correct amino acid appears in a particular column of a block, the mutation is possible to recover. This measurement can be compared to the number of extra mutations that are incorrectly assigned. We will also keep track of how many amino acids are verified by Spider tags that we use in the assembly. This measurement will be deemed coverage.

The testing of Champs was done by using two protein samples that were digested with three enzymes, GluC, LysC, and trypsin, creating many overlapping peptides as required. This process can be run on

a LTQ-Orbitrap hybrid instrument. The result spectra were sequenced via *de novo* sequencing on PEAKS 5.2 software and *de novo* sequences would only be used if they were both the highest scoring sequence for a given spectra and had a confidence score > 30%.

The proteins used were ALBU_BOVIN (Serum albumin precursor from Bos taurus (Cow)) and LYS_CHICK (Lysozyme C from Gallus gallus (Chicken)). During the process of testing, these proteins were removed from a SwissProt database and a new search is conducted using regular database search to find a homologous protein. ALBU_SHEEP (ALBU_BOVIN's counterpart from sheep) and LYSC_COTHA (Lysozyme C from Japanese quail) were picked as references respectively.

5.4 Results

The results are as displayed in Table 10 and Table 11 which also display the results of the algorithm at varying α values. We can verify that accuracy is upwards of 98% and 99% respectively while both proteins have 100% coverage. We can also see that the algorithm is relatively robust with little variation when varying α.

Total	Correct[27]	Found mutations[28]	Incorrect AAs	Extra mutations[29]	Missed mutations[30]	Correctable AAs[31]	Covered AAs[32]	Accuracy (%)[33]	Coverage (%)	α
583	558	23	25	2	23	22	583	95.7	100.0	1
583	570	35	13	3	10	10	583	97.8	100.0	2
583	573	38	10	3	7	7	583	98.3	100.0	3
583	573	38	10	3	7	7	583	98.3	100.0	4
583	572	38	11	4	7	8	583	98.1	100.0	5
583	573	39	10	4	6	7	583	98.3	100.0	6
583	573	39	10	4	6	7	583	98.3	100.0	7
583	573	39	10	4	6	7	583	98.3	100.0	8
583	573	39	10	4	6	7	583	98.3	100.0	9
583	573	39	10	4	6	7	583	98.3	100.0	10

Table 10: BSA dataset - variable α

[27] "Correct" refers to the number of columns that reconstructed the correct amino acid out of the total
[28] "Found mutations" refers to the number of mutations (from the reference) that we successfully reconstructed
[29] "Extra mutations" refers to the number of mutations that we falsely reported
[30] "Missed mutations" refers to the number of mutations that we did not successfully reconstruct
[31] "Correctable AAs" refers to the number of missed mutations where we had a chance of recovery
[32] "Covered AAs" and "Coverage" refers the number of amino acids which were matched to a Spider tag
[33] "Accuracy" refers to the portion of the protein that was successfully sequenced

Total	Correct	Found mutations	Incorrect AAs	Extra mutations	Missed mutations	Correctable AAs	Covered AAs	Accuracy (%)	Coverage (%)	α
129	126	3	3	0	3	3	129	97.7	100.0	1
129	127	4	2	0	2	2	129	98.4	100.0	2
129	127	4	2	0	2	2	129	98.4	100.0	3
129	128	5	1	1	0	1	129	99.2	100.0	4
129	128	5	1	1	0	1	129	99.2	100.0	5
129	128	5	1	1	0	1	129	99.2	100.0	6
129	128	5	1	1	0	1	129	99.2	100.0	7
129	128	5	1	1	0	1	129	99.2	100.0	8
129	128	5	1	1	0	1	129	99.2	100.0	9
129	128	5	1	1	0	1	129	99.2	100.0	10

Table 11: LysC dataset - variable α

Chapter 6
Conclusion

6.1 Author's Contributions

While several heuristic approximations of the Spider homology search algorithm have been previously implemented for solving the searching problem, this thesis contributes an algorithm, referred to as Spider II that unifies reconstruction with the search problem. This new algorithm also contributes a method for caching the calculation of intermediate results, thus reducing a combined polynomial runtime of the search algorithm from $O(m + |Z|^2 \times |\sum|) \times |X|^2)$ to a $O(|\sum|^9)$ runtime, independent from the data, which can be cached from run to run and a smaller polynomial $O(|X||Z|)$ search runtime. This separation also removed the dependency of the runtime on the number of variable PTMs chosen ($|\sum|$). This allows for a realistic runtime with variable PTMs while being comparable in performance with the most thorough variant of the previous version of Spider (which could not handle variable PTMs).

Several heuristics for investigating the selection of seeds for further investigation with the search algorithm were also investigated and found to be relatively robust.

Reconstruction has proven to provide a rich source of additional information performing significantly better than both the baseline *de novo* approach and an error-tolerance database search approach, delivering a significant number of exact matches and correct amino acids. Additionally, it has been shown that Spider's fairly unique approach delivers matches that are largely complimentary to the competing approaches.

6.2 Collaborative Aspects

As noted earlier, the original Spider algorithm was implemented and evaluated by Yonghua Han and subsequently integrated by BSI staff into PEAKS Studio. The author built on top of this framework to implement Spider II.

While the author was primarily responsible for the development of Spider II, the work on Champs was done in collaboration with Xiaowen Liu with Spider reconstruction allowing for the development of the Champs full protein sequencing application. Xiaowen Liu was responsible for the development and initial benchmarking of the Champs algorithm while the author was responsible for its interface with Spider II and subsequent evaluation.

6.3 Future Work

There are a number of possible avenues for future research. The performance of the new Spider II algorithm for search could use some improvement. One approach would be to try to fine-tune the heuristics that select possible seeds for further investigation.

There are also a number of choices in the equations used to model the alignment score. Some of these choices could be further explored in order to yield a better model. For example, experiments could be done with different BLOSUM matrices to see if it is worthwhile using different matrices when it is known how divergent the homologous database is from the original species.

As for the Champs algorithm for full protein sequencing, there are also a number of possible next steps. One possible step would be to develop a method for the characterization of PTMs. D*e novo* sequencing and Spider can identify PTMs, but the Champs algorithm strips out this information. This could potentially be improved for the purposes of identifying unexpected mutations. Another avenue of possible investigation would be to determine the efficacy of using Champs to verify the sequence of known proteins with high confidence rather than the characterization of novel proteins.

Appendix A
List of Software and Hardware Used

This section lists the software programs and hardware used.

Software List

- PEAKS Studio 5.2, PEAKS Studio 5.1, PEAKS Studio 4.5
    - *de novo* sequencing, database search and Spider homology search
- DirecTag 1.3.24 (2010-7-21)
- TagRecon 1.2.34 (2010-6-21)
- MASCOT 2.3.01
- Groovy 1.7.0
- Java 1.6.0_18

Hardware List

- Personal Computer: Windows 7 64-bit, Intel(R) Core(TM) i7 CPU 860 @ 2.80 GHz (8CPUs) with 8192MB RAM

Bibliography

[1] B. J. M. Webb-Robertson and W. R. Cannon, "Current trends in computational inference from mass spectrometry-based proteomics," *Briefings in Bioinformatics*, vol. 8, no. 5, p. 304, 2007.

[2] E. V. Koonin and M. Y. Galperin, *Sequence - Evolution - Function: Computational Approaches in Comparative Genomics*, 1st ed. Springer, 2002.

[3] Y. Han, B. Ma, and K. Zhang, "Spider: Software for Protein Identification from Sequence Tags with de novo Sequencing Error," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 3, p. 697, 2005.

[4] F. Crick, "Ideas on protein synthesis," in *Symp. Soc. Exp. Biol*, 1958, vol. 12, p. 63.

[5] L. A. Martin-Visscher et al., "Isolation and characterization of carnocyclin a, a novel circular bacteriocin produced by Carnobacterium maltaromaticum UAL307," *Applied and Environmental Microbiology*, vol. 74, no. 15, pp. 4756-4763, Aug. 2008.

[6] B. Ma, "Challenges in Computational Analysis of Mass Spectrometry Data for Proteomics," *Journal of Computer Science and Technology*, vol. 25, no. 1, pp. 107-123, Jan. 2010.

[7] P. Hernandez, M. Muller, and R. D. Appel, "Automated protein identification by tandem mass spectrometry: Issues and strategies," *Mass Spectrometry Reviews*, vol. 25, no. 2, pp. 235-254, 2006.

[8] Kkmurray, *File:Peptide fragmentation.gif - Wikimedia Commons*. .

[9] H. Steen and M. Mann, "The abc's (and xyz's) of peptide sequencing," *Nat Rev Mol Cell Biol*, vol. 5, no. 9, pp. 699-711, 2004.

[10] B. Ma et al., "PEAKS: powerful software for peptide de novo sequencing by tandem mass spectrometry," *Rapid Communications in Mass Spectrometry*, vol. 17, no. 20, pp. 2337-2342, 2003.

[11] A. Frank and P. Pevzner, "PepNovo: de novo peptide sequencing via probabilistic network modeling," *Anal. Chem*, vol. 77, no. 4, pp. 964-973, 2005.

[12] R. S. Johnson and J. A. Taylor, "Searching sequence databases via de novo peptide sequencing by tandem mass spectrometry," *Molecular Biotechnology*, vol. 22, no. 3, pp. 301-315, Nov. 2002.

[13] B. Fischer et al., "NovoHMM: a hidden Markov model for de novo peptide sequencing," *Anal. Chem*, vol. 77, no. 22, p. 7265–7273, 2005.

[14] D. L. Tabb, Z.-Q. Ma, D. B. Martin, A.-J. L. Ham, and M. C. Chambers, "DirecTag: Accurate Sequence Tags from Peptide MS/MS through Statistical Scoring," *Journal of Proteome Research*, vol. 7, no. 9, pp. 3838-3846, 2008.

[15] V. Dancik, T. A. Addona, K. R. Clauser, J. E. Vath, and P. A. Pevzner, "De novo peptide sequencing via tandem mass spectrometry," *Journal of Computational Biology*, vol. 6, no. 3-4, p. 327–342, 1999.

[16] L. Y. Geer et al., "Open mass spectrometry search algorithm," *Journal of proteome research*, vol. 3, no. 5, p. 958–964, 2004.

[17] D. N. Perkins, D. J. C. Pappin, D. M. Creasy, J. S. Cottrell, and others, "Probability-based protein identification by searching sequence databases using mass spectrometry data," *Electrophoresis*, vol. 20, no. 18, p. 3551–3567, 1999.

[18] D. Fenyö and R. C. Beavis, "A method for assessing the statistical significance of mass spectrometry-based protein identifications using general scoring schemes," *Analytical Chemistry*, vol. 75, no. 4, pp. 768-774, Feb. 2003.

[19] A. J. Link et al., "Direct analysis of protein complexes using mass spectrometry," *Nature Biotechnology*, vol. 17, no. 7, p. 676–682, 1999.

[20] D. M. Creasy and J. S. Cottrell, “Error tolerant searching of uninterpreted tandem mass spectrometry data,” *Proteomics*, vol. 2, no. 10, pp. 1426-1434, 2002.
[21] T. Dobzhansky, “Biology, Molecular and Organismic,” *American Zoologist*, vol. 4, no. 4, pp. 443-452, Nov. 1964.
[22] A. Shevchenko et al., “Charting the Proteomes of Organisms with Unsequenced Genomes by MALDI-Quadrupole Time-of-Flight Mass Spectrometry and BLAST Homology Searching,” *Analytical Chemistry*, vol. 73, no. 9, pp. 1917-1926, May. 2001.
[23] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403-410, Oct. 1990.
[24] S. Dasari, M. C. Chambers, R. J. C. Slebos, L. Zimmerman, A.-J. Ham, and D. L. Tabb, “TagRecon: high-throughput mutation identification through sequence tagging,” *Journal of Proteome Research*, no. ja, 0.
[25] B. C. Searle et al., “Identification of protein modifications using MS/MS de novo sequencing and the OpenSea alignment algorithm.,” *Journal of proteome research*, vol. 4, no. 2, p. 546.
[26] D. L. Tabb, A. Saraf, and J. R. Yates, “GutenTag: High-Throughput Sequence Tagging via an Empirically Derived Fragmentation Model,” *Analytical Chemistry*, vol. 75, no. 23, pp. 6415-6421, Dec. 2003.
[27] H. Yonghua, “SPIDER: software for protein identification from sequence tags with de novo sequencing errors.” [Online]. Available: http://www.csd.uwo.ca/Talks/TPJan20_100_04.shtml. [Accessed: 19-Feb-2010].
[28] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, p. 10915, 1992.
[29] S. Hopper, R. S. Johnson, J. E. Vath, and K. Biemann, “Glutaredoxin from rabbit bone marrow. Purification, characterization, and amino acid sequence determined by tandem mass spectrometry.,” *Journal of Biological Chemistry*, vol. 264, no. 34, pp. 20438-20447, Dec. 1989.
[30] N. Bandeira, K. R. Clauser, and P. A. Pevzner, “Shotgun protein sequencing: assembly of peptide tandem mass spectra from mixtures of modified proteins,” *Molecular & Cellular Proteomics: MCP*, vol. 6, no. 7, pp. 1123-1134, Jul. 2007.
[31] N. Bandeira, V. Pham, P. Pevzner, D. Arnott, and J. R. Lill, “Automated de novo protein sequencing of monoclonal antibodies,” *Nat Biotech*, vol. 26, no. 12, pp. 1336-1338, Dec. 2008.
[32] N. E. Castellana, V. Pham, D. Arnott, J. R. Lill, and V. Bafna, “Template proteogenomics: sequencing whole proteins using an imperfect database,” *Molecular & Cellular Proteomics*, vol. 9, no. 6, p. 1260, 2010.
[33] S. Tanner et al., “InsPecT: identification of posttranslationally modified peptides from tandem mass spectra,” *Anal. Chem*, vol. 77, no. 14, p. 4626–4639, 2005.
[34] “File:Needleman-wunsch.jpg - Wikimedia Commons.” [Online]. Available: http://commons.wikimedia.org/wiki/File:Needleman-wunsch.jpg?uselang=en-gb. [Accessed: 21-Feb-2011].
[35] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins,” *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443-453, Mar. 1970.
[36] S. F. I. Pevtsov, H. B. C. Mirzaei, and X. Zhang, “Performance evaluation of existing de novo sequencing algorithms,” *Journal of proteome research*, vol. 5, no. 11, p. 3018–3028, 2006.
[37] M. Bin and D. Yuen, “Novel Scoring Function Improves Homology Searches Using MS/MS de novo Sequencing Results,” presented at the ASMS, 2008.
[38] P. C. Andrews et al., “ABRF-sPRG2006 Study: A Proteomics Standard,” *Proteins*, vol. 5, no. 48, p. 645.

[39] L. Mo, D. Dutta, Y. Wan, and T. Chen, "MSNovo: A Dynamic Programming Algorithm for de Novo Peptide Sequencing via Tandem Mass Spectrometry," *Analytical Chemistry*, vol. 79, no. 13, pp. 4870-4878, Jul. 2007.

[40] J. Morey, I. Rogers, and C. Chen, "Filtering out MS/MS spectra of insufficient quality before database searching," *PEAKS Proteomics Research*. [Online]. Available: http://www.bioinformaticssolutions.com/products/peaks/db_bsipaper.php. [Accessed: 12-Mar-2011].

[41] X. Liu, Y. Han, D. Yuen, and B. Ma, "Automated protein (re) sequencing with MS/MS and a homologous database yields almost full coverage and accuracy," *Bioinformatics*, vol. 25, no. 17, p. 2174, 2009.