An Investigation of Protein Database Search Parameters Using Monte Carlo Methods

by

Eric Van Uytven

A Thesis

Presented to the Faculty of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree of

Master of Science

Department of Physics

University of Manitoba

Winnipeg, Manitoba, Canada

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

AN INVESTIGATION OF PROTEIN DATABASE SEARCH PARAMETERS USING MONTE CARLO METHODS

BY

ERIC VAN UYTVEN

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree of

Master of Science

Copyright Declaration

ABSTRACT

Protein sequence identification by mass spectrometric (MS) peptide mapping is an important tool in proteomics research. This method of protein identification has become the tool of choice for identification of unknown proteins. Challenges exist in improving the performance of these protein database searches. Many search engines exist, both public and proprietary. Underlying algorithms are often different in many ways. Improving these powerful tools consists mostly of developing algorithms to provide accurate and unambiguous protein matches, as well as to account for various protein and peptide modifications that can occur in experimental work. One specific area that is being investigated is the coding of more complex algorithms to eliminate false results in protein database searches due to coincidence matching. It is often the case that multiple matches may be returned in a single search.

A method is developed of performing many Monte Carlo simulations of protein database searches while iterating through various search parameters. The number of false matches returned is recorded in each case, and the 'parameter space' of protein database searches is mapped out. By examining the parameter space, one can investigate how the occurrences of false matches are correlated with search parameters, and ultimately learn how to improve protein database search algorithms.

Certain combinations of search parameter restrictions were found to reduce and eliminate false matches. The most important of these parameters is the number of required peptide matches, with a 10-fold reduction in false matches per monotonic increase of rpm. 4 rpm eliminates false matches at 20ppm, 5 rpm at 35ppm, and 6ppm at

75 ppm. False matches vary with mass accuracy is a sigmoid fashion, with 0 false matches at 0ppm, 1 false match at 5ppm and increasing to 70 false matches at 100ppm.

ACKNOWLEDGEMENTS

I would like to express my thanks to my co-supervisors, Werner Ens and Ron Beavis, who helped me overcome difficulties and challenges that I would not have been able to surmount by myself. I would also like to thank the other members of my examining committee, Dr. John Wilkins, and the head of my laboratory, Dr. Ken Standing, who generously allowed me the use of lab computers.

Special thanks go to Vic Spicer, whose constant presence and tireless assistance to my project and myself was much appreciated.

Finally, I would like to thank my parents, Guy and Renee, for their loving support and encouragement over the years.

TABLE OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

LIST OF ACRONYMS

MA -	Mass Accuracy
nRMP -	Number of required matching peptides
MC -	Missed Cleavages
FMPPM -	False matches per peptide map
MINAA -	Minimum number of amino acids contained in a peptide
PMDS -	Peptide Mapping Database Search
PDB -	Protein Databank
PDSE -	Protein Database Search Engine
TDB -	Target Database
RPM -	Random Peptide Map
PMW -	Protein Molecular Weight
PDS -	Protein Database Search

Chapter 1

Bioinformatics

1.1 A short introduction

In 1857, Gregor Mendel performed an experiment involving thousands of pea plants and started the field of genetics. During the 20th century, research in genetics increased and microscope technology offered tantalizing looks at the microbiological world. In the early 1900's it was becoming accepted that the long strings of nucleic acids and protein contained within the nucleus, called chromosomes, were the storage units of hereditary information. In 1953, Watson and Crick developed a correct three-dimensional model of the DNA molecule [12].

At present, there is a basic understanding of how genes code for proteins. It is known that certain combinations of nucleic acids code for certain amino acids. Strings of these amino acids form proteins. There exists a rule set that dictates how nucleic acids translate to proteins, which scientists have not as yet fully explored.

Every cell in the human body contains 46 chromosomes, containing approximately 30,000 genes[1]. These genes contain about 3.2 billion nucleotide base

[1] A gene is a hereditary unit consisting of a sequence of DNA that occupies a specific location on a chromosome and determines a particular characteristic in an organism.

pairs[2]. A means of collecting, storing, sorting, categorizing, and correlating this information is needed if it is to be useful for data mining and information retrieval.

Information technology and data mining techniques have been developed with the goal of analyzing raw biological data such as the peptide masses obtained by mass spectrometry of biomolecules. There is a substantial interest in sequencing both the genome and proteome of various organisms. While genetics has existed for over a century proteomics - the sub-field of analysis and characterization of proteins, has recently received substantial growth. The goal of research in both fields is improved diagnosis, understanding, and treatment of disease as well as the understanding of biological systems. Looking for patterns in the sequencing information allows better drug development.

The need to assemble and correlate a wealth of biological data requires the development of computational tools. The use of computers in solving biological information problems is called Bioinformatics. Bioinformatics is a curious melding of computer science and biology, and in the past has called computational biology. Bioinformatics serves the scientific community by allowing scientists to use computational tools such as to:

1. Discover Normal Biological Processes
2. Understand abnormal Biological Processes
3. Facilitate Drug Discovery

Examples of the use of bioinformatics include database construction and management as well as 3-D modeling of biomolecules.

Bioinformatics enables accurate and efficient mapping of the genome and proteome.

---

[2] The pair of nitrogenous bases, consisting of a purine linked by hydrogen bonds to a pyrimidine, that connects the complementary strands of DNA or of hybrid molecules joining DNA and RNA. The base pairs are adenine-thymine and guanine-cytosine.

The proteome is the end result of the genome expression process. Activity and development in bioinformatics occurs both in the public and private sectors, developing both public and proprietary software tools to assist scientists in analyzing the data they have collected.

1.2 Nucleotide and Protein Databases

In 1982 nucleotide sequence databases were introduced starting with EMBL, the European Molecular Biology Laboratory [3]. EMBL was soon joined by and linked to the Genbank database in the United States, hosted at the National Center for Biotechnology Information (NCBI)[4], and the DNA Database of Japan [5] (DDBJ). During the 1990's, these database centers were collecting published sequence data and transcribing them to computer format. The rapid growth of the EMBL/Genbank databases is shown in fig. (1.1). Currently all databanks accept submissions.

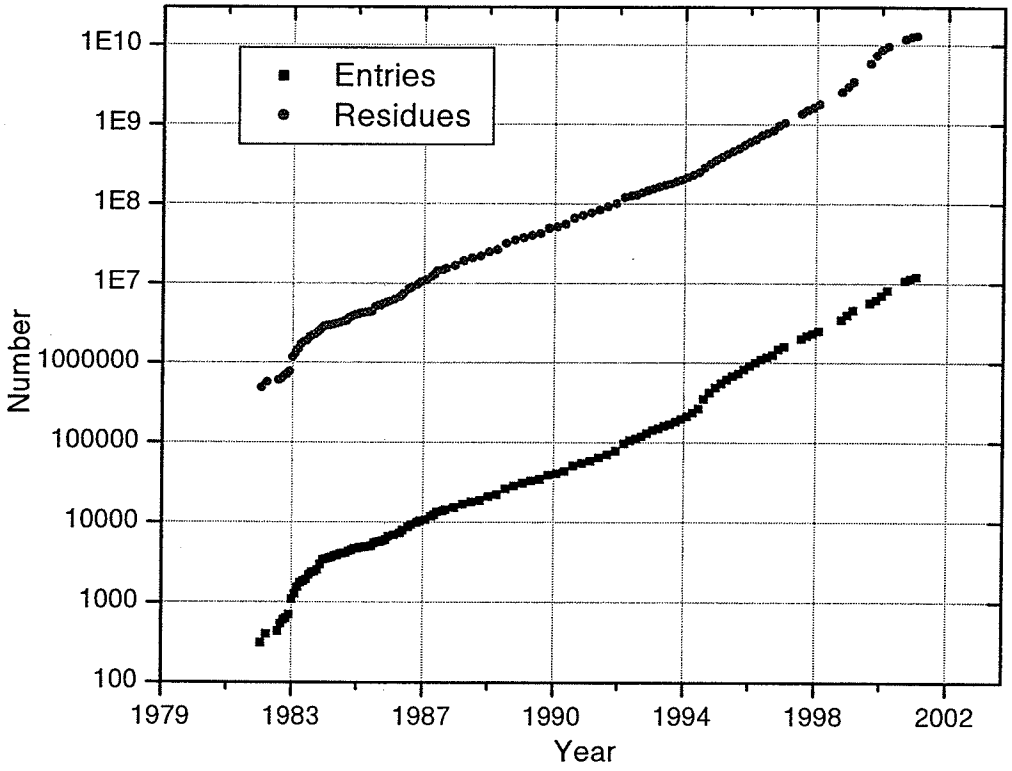

Figure 1.1: Growth of EMBL/Genbank

The three major databank centers agreed to a common format for exchanging sequence information, as well as to updating each other with new submission information on a daily basis. However, the file format that each database uses locally is slightly different. The foundation for a working protein database goes back as early as 1965. During the period 1965-1978 the *Atlas of Protein Sequences and Structures* was published under the editorship of Margaret O. Dayhoff [13]. Dayhoff and her group worked on computer algorithms designed to perform rudimentary tasks on a protein database, including protein sequence comparison, homology[3] studies, and detections of various sequence characteristics.. By 1981 a rudimentary protein database as well as computer algorithms for search and retrieval of sequence information was in place. This database was called

the PIR database, or Protein Information Resource [1]. The database still exists today, working in collaboration with NBRF (National Biomedical Research Foundation), The Munich Information Center for Protein Sequences (MIPS), and the Japan International Protein Information Database (JPID).

PIR is one of the major protein databases in the world, producing the most comprehensive and expertly annotated database in the public domain [24]. Other databases include the Swiss-Prot database and the structure-oriented Protein Databank (PDB). The PIR and Swiss-Prot are curated sequence databases[4], while the PDB is a structure database, containing a limited amount of proteins with fully mapped 3D structural information. Presently, these protein databases derive most of their protein sequence information from nucleotide sequences. In 1990, software was developed to amalgamate sequence entries from PIR, Swiss-Prot, and GenBank (translation) libraries. The software, which fully automated the process, would eliminate redundant and near-redundant entries to provide fully non-redundant composite database [14]. The OWL protein database [2] is one such example.

These three nucleotide and protein databases contain extensively annotated(additional information above and beyond the sequence) sequence entries. Three-databank records are illustrated in appendix A. These databanks have developed into centers with a host of bioinformatics development activity. The protein and nucleotide databases are constantly reviewed and updated by biologists, and tools are constantly being developed to further data handling and ease of retrieval. Some examples of these tools are Entrez/BLAST/ORF(open reading frame)[5] finder from NCBI, and SRS(Sequence Retrieval System) being developed at the EBI.

---

[3] Structures or traits that may have different functions in different organisms but which originated from a common ancestor.

[4] The database is actively maintained and sequence submissions are scrutinized to ensure non-redundant entries in the database.

[5] Entrez is an article search engine. BLAST is a sequence search engine. The ORF finder locates open reading frames, which are sequence areas in genes that code for proteins.

1.3 Database Search Engines

An enzymatic digest is a manner in which proteins can be broken down in a controlled manner. An enzyme such as trypsin is introduced in a solution with a protein mixture; the enzyme will break down specific amino acids bonds. Trypsin cleaves with high specificity at the carboxyl side of lysine and arginine residues. Other enzymatic digest agents exist such as chymotrypsin and V-8 protease, each with its own specific pattern of cleaving. The peptide fragments can be run in a mass spectrometer to provide a peptide mass 'fingerprint'. These peptide mass fingerprints have become the method of choice for rapid identification of sample proteins.

Many of the scientists working in genomics and proteomics are interested in using the tools of bioinformatics for protein identification. Consequently, a large amount of bioinformatics development activity is concentrated on sequence identification and retrieval [25]. Specifically, protein identification by peptide mapping using private and public domain 'search engines' allow the identification of a protein using experimental peptide masses. The most common type of this search is protein identification using peptide-map fingerprint data, which postulates that a list of peptide masses obtained by mass spectrometry or other means from an unknown protein is unique. Finding peptide mass matches between the unknown protein and a protein within a sequence database can identify the parent protein.

Peptide masses are best obtained by mass spectrometry. However, until about a decade ago, mass spectrometric techniques were only useful with small biomolecules. The main problem for MS was the necessary ionization of the sample. Older techniques were unable to ionize peptides and samples containing nucleic acids without fragmenting the sample. However, within the last decade, so-called "soft" ionization techniques were

developed to overcome this problem. Specifically, the development of techniques such as Matrix-Assisted Laser Desorption and Ionization (MALDI)[18][19] and ElectroSpray Ionization (ESI)[20] permitted the use of mass spectrometric analysis to a large range of peptides. Using such ionization techniques coupled to a good mass spectrometer (ESI-MS and MALDI-MS), one can confidently obtain a set of high accuracy peptide masses.

Proper identification requires that the unknown protein data come from an organism with a sequenced genome. Most of today's protein databases contain annotated entries of tens of thousands of proteins from many organisms. These databases contain complete molecular weight information of proteins as well as a calculated molecular peptide map. Protein database searches use these maps to enable identification of experimental unknown peptide maps. Protein database search engines are available on the internet (Profound/MS-TAG/MOWSE)[6]. There are also software packages available to run searches on local protein databases.

Experimenters typically have success when mapping simple protein samples belonging to highly sequenced genomes. However, in cases where no clear match is found, such as is found when mapping against a complicated partially sequenced genome, more experimental information is required.

In the event of ambiguous or otherwise problematic identifications, tandem mass spectrometry (MS/MS) is usually the next step. Certain peptides are isolated and fragmented into smaller peptides. Protein databases can predict the masses of fragment ions; typically, the masses of all possible fragments are calculated and compared to the experimental data. A score is then calculated. Other MS/MS methods are being developed to reduce the problems with MS/MS as well as increasing the throughput. [28]

Typically, identification of a protein follows these steps:

---

[6] -ProFound (dev. By Proteometrics): http://prowl.rockefeller.edu/cgi-bin/ProFound
-MS-TAG (dev. By UCSF): http://prospector.ucsf.edu/
MOWSE (dev. By UK HGMP): http://www.hgmp.mrc.ac.uk/Bioinformatics/Webapp/mowse/

1. Separation of a protein mixture using 1D or 2D electrophoresis
2. Performing an enzymatic digest on a protein of interest (an enzyme such as trypsin will cut proteins into peptides at known locations)
3. Ascertaining the mass of the tryptic fragments by mass spectrometry
4. Comparing to protein database sequences.
5. Performing MS/MS in the event of an ambiguous match

The process is illustrated in figure (1.2). Once the mixture is separated each separate band is cut out and its corresponding protein identified.

Additional specificity can be obtained by sorting the matches based on additional parameters. Some of these include:

- Protein isoelectric point (estimated from a 2-D gel)
- Protein molecular mass (estimated from a 2-D gel or by MS)
- Species of origin of the protein
- Protein amino acid composition
- Protein sequence data (sequence tag)

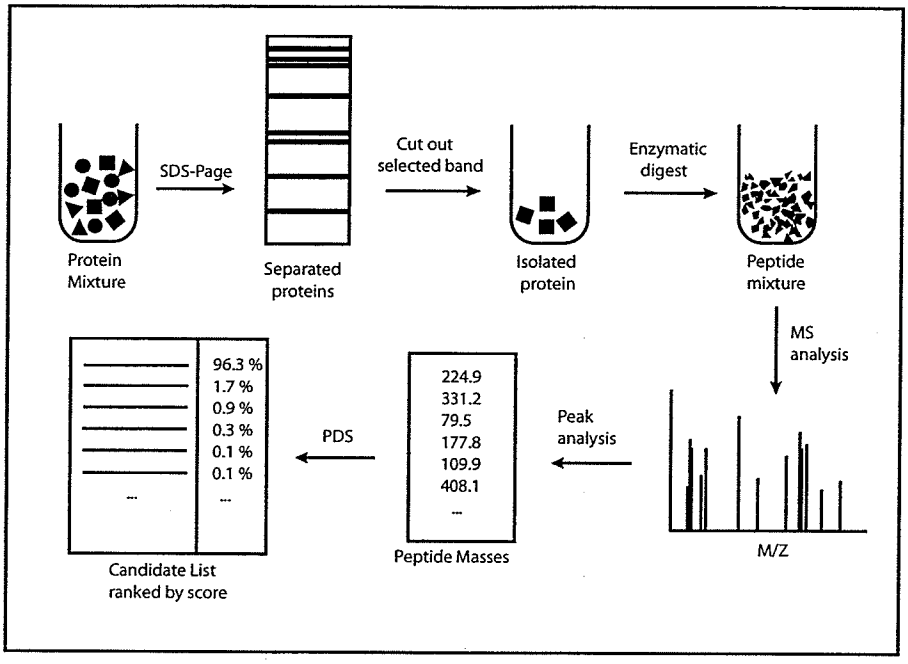

Figure 1.2: Identification of unknown proteins in a mixture

There are several ways that algorithms calculate the resultant matches and scores associated with those matches. Early algorithms calculated a score based directly on the number of matching peptides (peptide masses that match within a certain predefined uncertainty). Current algorithms use many different kinds of statistical analysis to return a correct candidate using input data, which is often not enough to isolate a correct match based simply on number of matching peptides.

MOWSE

MOWSE or Molecular Weight Search, hosted at UK Human Genome Resource Center[6] is one of the first protein database search engines. Designed and implemented in the early 1990's, the implementation was a database of molecular weight fragments,

combined with search and scoring algorithms for use with peptide mass fingerprints as input.

The database was constructed from information derived from the non-redundant OWL database. Each parent sequence of OWL was theoretically digested using several kinds of enzymes (trypsin, endoprotease Lys-C, chymotrypsin, etc...) and the resultant peptide masses were calculated and stored with the parent sequence in the database.

A user can then input a peptide mass fingerprint, comprising a list of peptide masses with associated mass error; MOWSE will query the database and return a list of candidate proteins, ranked not only by number of peptide 'hits' per candidate, but also with a weighting factor that is dependant on the frequency of that particular fragment being found in all proteins of mass similar to that of the candidate [16]. MOWSE can also narrow the results provided the experimenter knows the approximate molecular weight of the parent protein.

Closely related to MOWSE is Mascot [7], which uses the MOWSE engine, but also handles matches on a probabilistic basis, calculated on the probability that a match results are the result of random matching.

MS-FIT

MS-Fit from the UCFS Mass Spectrometry Facilities is a typical example of a search engine which returns results ranked by the number of peptide matches. MS-Tag is a tandem mass spectrometry search engine that works in a similar fashion. These tools are part of the Protein Prospector package [8]. Similar algorithms are used for PepSea, Multident at the Swiss Institute of Bioinformatics [9], and Pepfrag [10], another MS/MS engine hosted on the Proteometrics website.

PROFOUND

The Profound search engine uses a matching system based on Bayes' theorem [23]. Shown in left hand side of figure (1.3) is a simple search window of the Profound search engine. The default database that is searched is the non-redundant protein database at NCBI. This database compromises all non-redundant GenBank CDS translations+PDB+SwissProt+PIR. The taxonomic category allows the user to limit the search within a particular organism. The default enzymatic digest is trypsin, however it is possible to search theoretical peptide maps digested using such enzymes as V8 and endoproteinase Lys-C.

The list of experimental peptide masses is input at the bottom of the screen. The masses may either be monoisotopic masses (lowest mass isotope) or average masses (average mass of all isotopes). The uncertainty in the masses of the peptides is another input parameter. The default number of missed cleavages is 1. Missed cleavages indicate a possibility that trypsin (or whichever enzyme was used) did not cut the protein at all cleavage points. 1 missed cleavage indicates that at most, the enzyme missed one cleavage per fragment scored. It is not known a priori how many cleavages the digestive agent may have missed. By setting this parameter higher more possible peptide matches are covered, however there is a high incidence of false matches as well. The right hand frame of figure (1.3) illustrates the result summary of the search.

The candidate proteins are ranked according to a calculated expectation value, which is the probability of scoring better using a random protein sequence.

Figure (1.4) is the ProFound window that provides details about one of the candidate proteins. The first three graphs illustrate the number of matching peptides, as well as the extent of which they cover the sequence of the candidate protein. A table in

the bottom of the window includes details about each separate matching peptide. Sequence coverage graphs as well as a table illustrate the details of the peptide matches.

In many cases the correct protein may not be ranked first in the list of potential candidates; this is when it becomes an arduous task to find the correct match among a large amount of false matches. Indeed, the small differences in the algorithms of protein database search engines can mean the difference between unambiguous identification of a protein and a hopeless list of candidates.

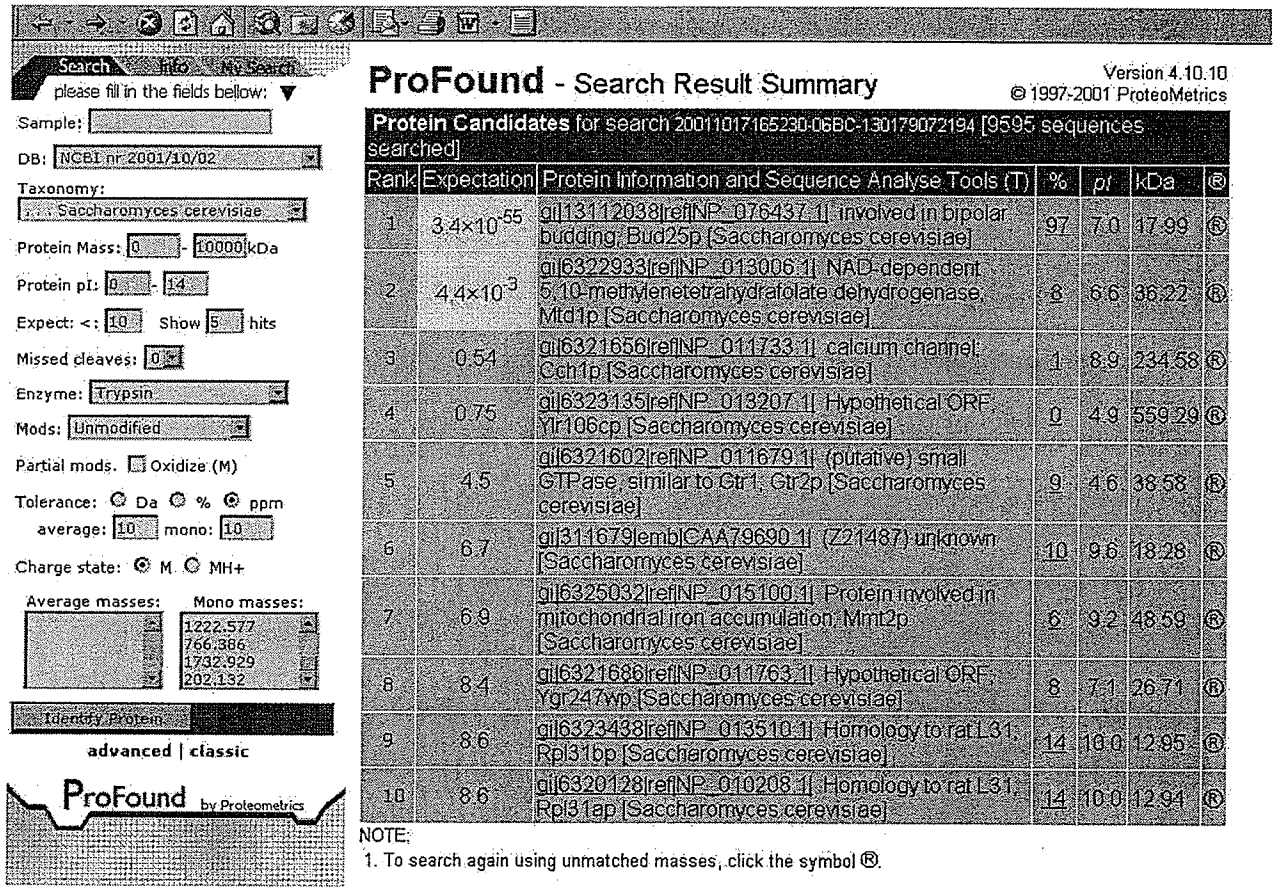

Figure 1.3: Profound - Search Results of a Profound Database Search. Ideal peptide masses were used in the search to illustrate an unambiguous identification.

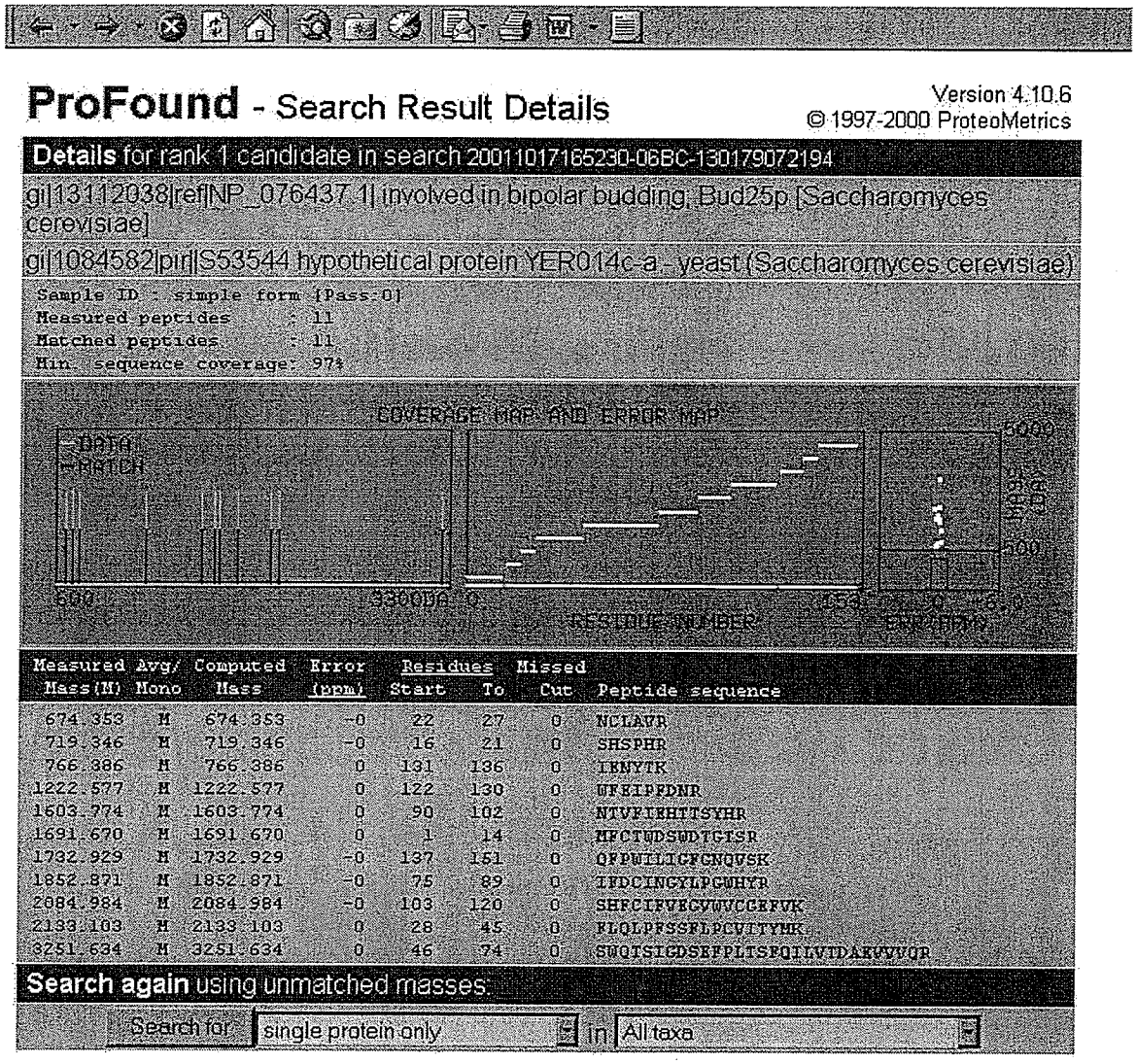

Figure 1.4: Profound - Detailed results of one of the candidate proteins.

Chapter 2

Research Problem and Methodology

2.1 Overview

Thesis goals:

1. **Generation of random peptides to be used as source data for PDB searches**
2. **Exploration of parameter space, variation and analysis of:**
    - *Mass accuracy*
    - *Number of peptides in a PDB match*
    - *Random peptide map size*
    - *Number of missed cleavages*
    - *Random peptide size*
    - *Molecular weight range*
    - *Parent Molecular Weight Range*
3. **Development of a cluster computer**
4. **Development of a generalized search engine that can be adapted to future applications.**

Accomplishments of these goals will be outlined and discussed in Chapter 5: Conclusions and future work.

In this chapter the research problem is discussed, and the methodology is covered. Section 3.2 discusses the programming model and section 3.3 details some of the in house cluster computing work accomplished to enable the thesis work.

2.2 Research Objectives

A large part of Bioinformatics is the development of protein databases and their associated search engines. Since this is a relatively new field there has not been extensive work done in examining statistical factors involved in these database searches.

Challenges exist in the area of peptide map database searches. Factors such as uncertainty in the experimental peptide masses, inability of protein molecules to be cleaved completely by trypsin, and non-unique theoretical peptide masses[7] lead to multiple 'hits', where it is not possible to distinguish between the correct protein and an unwanted list of 'false matches'. These 'false' matches are due to random coincidence matching of one or more peptides of the input peptide map with peptides contained in database proteins.

In order to eliminate these problems, we need to know the quality level of data that will lead to an unambiguous identification. The development of more sensitive and higher resolution mass spectrometers have allowed for increasingly more accurate peptide maps [21][22][27]. Specialized techniques such as chemically modified peptide maps [30] and high-accuracy approaches [29] have been successful in improving the

[7] It is possible for multiple proteins to contain peptides with amino acid configurations that have exactly or nearly exactly the same mass

quality of protein identifications. Equally important are developments in the bioinformatics tools used by experimenters to allow unambiguous identifications.

One of the major problems in protein database programming is writing algorithms that can separate high relevance database search matches from the 'false matches'. A false match occurs when the score (calculated in various ways depending on the algorithm) of the correct protein match is less than the score of various other matches that are due to random peptide matching. New algorithms are constantly being developed to overcome this problem.

As discussed in the previous chapter a number of methods have been developed to sort the search results by relevance. These methods use existing data provided by the user in a PDS to optimize the relevance of the search results.

This thesis uses a Monte Carlo method to determine the number of false matches in a protein database searches. A custom protein database search engine was developed that is both optimized for speed and iterative ease. By performing protein database searches while iterating over various search parameters, a 'parameter space' can be built up. Analysis can then be performed on the parameter space to elucidate which combinations of search parameters lead to a reduction in false matches. Relationships between certain search parameters can be exploited to help increase the accuracy of database searches. The results can also give understanding to the particular qualities of a peptide map that create ambiguous results.

The information collected during this project can also lead to the improvement of scoring algorithms by using results to alter weighting factors in standard database scoring algorithms. Since this thesis is interested in exploring the pattern of false matches over various search parameters, all database searches are performed with random tryptic peptide maps generated from a genome derived proteome database. Since the majority of

database searches are performed with tryptic digests, using random tryptic maps will ensure that the results are relevant.

The parameter space will be built up by varying the various search parameters detailed in section 2.1.

These parameters will be explained in detail in chapter 4.

2.3 Programming Challenges

The program needs to iterate over hundreds of possible parameter combinations, requiring that the search engine be programmed to automatically collect and store results of each iteration as well as starting a new one. The protein database searches need to be fast, as processor and time constraints would limit the possibility of performing searches against large databases. Additionally, random peptides need to be generated that have the properties of real peptides, such as approximate size, allowable amino acid chains, and correct start and end residues for a peptide generated by a particular enzyme.

Varying the parameter 'minimum number of matching peptides' is a study in the effects of reducing or increasing the number of required matching peptides, and required a generalized algorithm.

2.4 Processor Issues

Performing protein database searches requiring one or perhaps two matching peptides is an easy task for most modern computers. However, for 3 or more matching peptides, a challenge arises to write a suitably efficient and fast algorithm. The algorithm

needs to accommodate an arbitrary number of matching peptides, as well as being quickly executable since the algorithm will be called tens of thousands of times for a particular set of search parameters.

For example, consider a set of 200 peptide maps, each containing 30 peptides. In a PDS, each peptide in the set will need to be compared with each proteome peptide. A PDS run using fairly typical search parameters might run for approximately 20 minutes on a 500MHz Intel Pentium 2 platform running Windows 98SE. The proposed project involves doing similar runs thousands of times over. Due to the computational power required to complete the objectives of the project, one of the supplemental objectives became to enable a large throughput of calculations. The first approach to this problem was to minimize the number of floating point operations for one protein database search. However, due to the large volume of required calculations, the available solution was an increase in overall processor power. One computer would not be sufficient; some sort of parallel processing unit was indicated.

A workstation cluster[8] is similar to a parallel processing machine, but consists of discrete workstation computers, usually connected through a local area network. With decreasing cost and improving performance, workstation clusters are becoming ideal for high performance processing [15].

A challenge arose to convert several desktop machines into a cluster unit, which could advance the data collection rate to an appropriate level. Furthermore, since the desktop machines being considered were being used in other applications, the program needed to be run innocuously, and without consuming the bulk of any one computer's resources.

[8] Sometimes called "A poor man's supercomputer"

Chapter 3

Development

In this chapter the development of the code as well several research problems are discussed.

3.1 Overview

Figure (3.1) illustrates the relationships between classes of the pepstat program. The classes that are illustrated are the major working classes and threads. For the sake of brevity several smaller classes were left out. The diagram is illustrated in UML (Unified Modeling Language). A simple line indicates association between classes. A solid diamond line runs from a parent class to a daughter class that relies on the existence of the parent. A hollow diamond line is similar to the black diamond line, except the daughter can exist even when the parent doesn't. 1 .. * indicates that there is at least one (possibly more) daughter classes.

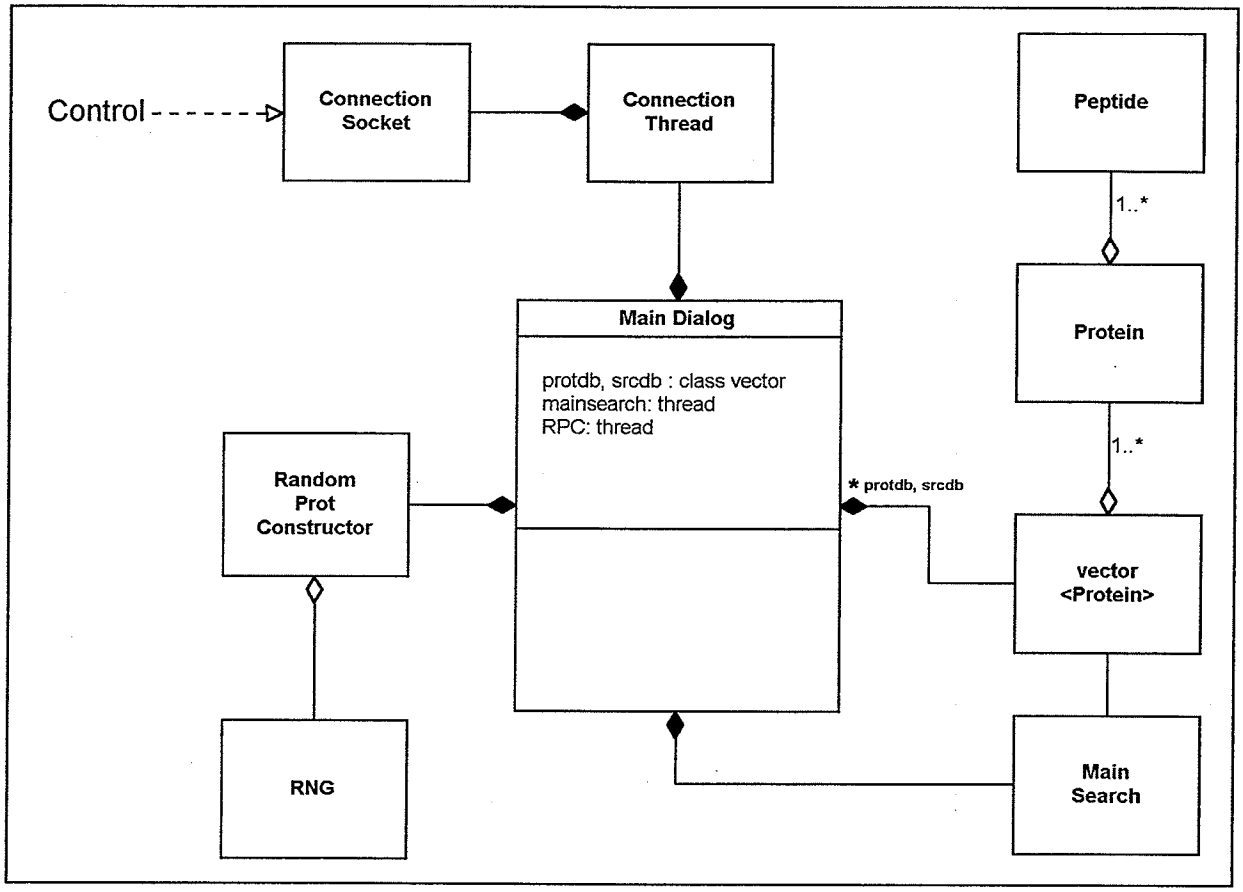

Figure 3.1: Pepstat Class Relationships

Each box represents a class or a thread. Descriptions of the various classes are as follows:

Main Dialog: This is both the core class of the program and the class that handles and displays the user interface dialog window.

MainSearch: This algorithm is a low priority worker thread[9], MainSearch performs the iterative matching between the peptide maps and the DDB.

Vector<protein>: Contained in the dialog class, a vector that allows for dynamic storage of n proteins. Each database (source and target) is a vector of type *Protein*.

Protein: A class that contains the characteristics of proteins. Attributes include average and monoisotopic mass and the protein sequence. A vector<Peptide> inside the protein allows for dynamic allocation of peptides. Methods in the class include those used for retrieving peptide sequences, calculating both peptide and protein mass, creating and deleting peptides among others.

Peptide: A class containing many of the attributes and methods of the protein class. This is the base unit of the databases.

Random Prot Constructor: A standalone thread executable from the dialog window. Accesses proteome files and generates customizable peptide maps.

RNG: Random Number Generator. Discussed in section 4.5.2.

Connection Thread: Generated following a connection attempt by a remotely located control program (see 4.9). The purpose of the thread is to allow connections from the control without disrupting program operations, and runs concurrently with the *Mainsearch* algorithm.

[9] A thread is a portion of a program that can run independently of and concurrently with other portions of the program. Threads can be instructed to run with various priorities. MainSearch is designed to run on

MainSearch algorithm: This allows for uninterrupted functioning of pepstat and other programs while large data files are being transferred.

ConnectionSocket: The class that handles TCP/IP communications with the control program. Communicates directly with the dialog to set parameters in the pepstat dialog window and automatically launch new protein searches.

Screenshots of the pepstat and control GUI's are listed in appendix C.

3.2 Local Databases

Given the limited computational power available, it was not feasible to perform protein database searches on a group of proteomes, as is the case when making a database search on such databases as OWL or Swiss-Prot. Performing iterative searches on one proteome using internet search algorithms would be impossible due to internet traffic and slow response time of any internet DBS engine.

Three proteomes were selected as local databases. All PDS searches were performed on these locally stored databases. The proteomes were selected based on certain criteria.

Proteomes vary widely based on taxonomic category they originate from. Thus the three selected proteomes selected had to be fundamentally dissimilar. However, together they had to be representative of the whole set of existing proteomes. If the false

---

IDLE priority status; it only runs during computing idle cycles. This ensures that pepstat would not interfere with computers that were running other high-priority processes.

match counts were similar between proteomes for a particular set of search parameters, then one could extrapolate the results to any desired proteome.

A desirable characteristic of a chosen proteome is the fact that it should be complete, or near complete. Any proteome with large gaps in the sequence data would not make a good candidate for the project work. Thus, the proteomes selected would be those, which are 'template' proteomes that have been extensively studied and sequenced.

These proteomes were also selected bases on size, ranging from 4400 to 18500 proteins in size. *E. coli* and *S. cerevisiae* are approximately the same size, however they are fundamentally dissimilar as one is a eukaryote and the other a prokaryote. *C. elegans* is a large multi-cellular eukaryote and was selected for that reason.

The following table describes the proteomes that were selected, some general characteristics, and the proteome size:

Escherichia coli	-*E. coli* is a gram-negative prokaryote bacteria -More known about *E. coli* than any other organism -Originally isolated in 1922 from a diphtheria patient, the strain of *E. coli* sequenced was used in 1945 in the discovery of spontaneous gene transfer. The strain, known as K-12, was universally adopted for fundamental work in biochemistry, genetics and physiology. -In recent years, it has become the 'heavy lifter' of biotechnology where it is used to produce human insulin and other medicines. -Fully Sequenced - ~ 4400 proteins

Saccharomyces cerevisiae "Brewer's yeast"	-Unicellular eukaryote organism -Used for millennia for it's role in food preservation and preparation - A simple, unicellular eukaryote developed to a unique powerful model system for biological research. - Fully Sequenced - ~6500 proteins

Caenorhabditis elegans	-*C. elegans* is a nematode, a complex multicellular organism. -Studied worldwide by hundreds of scientists -*C. elegans* is about as primitive an organism that exists which shares a substantial homology and similar reproductive cycle with humans. -All 959 somatic cells of its transparent body are visible with a microscope, and its average life span is a mere 2-3 weeks. Thus *C. elegans* provides the researcher with the ideal compromise between complexity and tractability. - Fully Sequenced - ~18500 proteins

The proteomes are downloaded from the Proteometrics web site and were used with permission. The file format for these proteomes is *bioml*, a file format designed to work with proprietary software also developed at Proteometrics. The files were stripped of any non-sequence information, such as labels, tags, and descriptors. Each protein sequence was placed in a new file, with one protein sequence per line. The number of proteins in each file was recorded and stored for later reference.

3.3 Generating Random Peptide Maps

3.3.1 Methodology

A method for generating non-existent random proteins was considered. These peptide maps need to be able to have pre-defined numbers of peptides. By performing search with these false proteins we ensure that all results are purely coincidental. With ideal search parameters false matches are approximately 0 due to a high degree of peptide mass uniqueness in peptides of ~5 amino acids or more.

Generating random tryptic peptides by stringing random amino acids together was briefly considered, but there are problems with this approach. One can examine basic characteristics of peptides, such as the average populations of amino acids, and average peptide length, however it would be impossible to randomly generate 'legitimate' peptides that could come from a biomolecule. Proteins fold in a distinct manner based on their amino acid sequence, and certain amino acids sequences are impossible to fold and hence don't exist in nature. As mentioned before, the generation process would need a rule set, which as of yet has not fully been constructed.

It was decided to construct random peptide maps by assembling peptides chosen randomly from a comprehensive list of proteomes. 15 non-redundant files were downloaded from the online proteome database at http://www.expasy.ch. The downloaded files are in FASTA[10] format. Each file contained proteome information for particular category of proteomes (for instance, eukaryotes, prokaryotes, etc...).

The files were then transcribed to 88 smaller files. All extraneous data was stripped out of the original files. This included all non-sequence information such as label, accession number, comments, etc.. The format of each of the 88 in house files is simply one protein sequence per line, preceded by a waka:

...

>RTRIQKHIKQAEASFIGHINPEHSNEQASTSLLSSSCHADHAVESYSSS...
>MDKKPCNSQDVE
>RTRIQKHMEQGDQSSSTTFNNGQMNLDHSCNDQASSSQMSACGPV
...

The waka (>) character allows indexing by the program. Each file contains 5000 proteins, except the last, which contains 3880. There were 433880 proteins in total. The algorithm for generating the random peptide maps is shown in figure (3.2). In this case the algorithm is generating 200 peptide maps.

[10] A sequence in FASTA format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. It is recommended that all lines of text be shorter than 80 characters in length.

```
create a string variable (protstring)
clear protdb
for (protindex=0; protindex<200, protindex++)
        -create a blank class(protein)
        -for (pepindex=0, pepindex<mapsize, pepindex++)
                -create a blank peptide inside the protein class
                -fill the blank peptide with a random peptide fragment using
                GETRANDPEPSEQ
                -add the peptide sequence to the end of protstring
        end for
        fill in the protein sequence(inside the class) using protstring; calculate masses.
end for
```

Figure 3.2: ProteinConstruct Class

The selection method is illustrated in figure (3.3). First a random protein is selected. From this random protein, a random peptide is selected. If the peptide is not complete[11] ('X' amino acids present in the peptide) the process is repeated. If the peptide size was less than a desired size, the process was again repeated. The selection process continues until the random peptide map is complete. 200 peptide maps were constructed and grouped into sets. These groups were used in the simulations.

[11] Not all proteomes are completely sequenced. This leads to gaps in protein sequences which are filled with placeholder amino acids: 'X'

---

index data files
select a random number from 0 to 433880 (the size of database containing all proteomes)
if (the random number is inside the designated range of the current target proteome)
    go back and re-select a number
determine which file the selected protein is in.
open the file and scan to the appropriate location
read the sequence of the selected protein
perform theoretical digest and count the number of resulting peptides
select a random number between 0 and the number of peptides
verify the peptide 'integrity'(peptides with X amino acids are discarded and the selection process is repeated from the beginning)
return the sequence of the selected peptide

Figure 3.3: GetRandPepSeq Class

There is one more problem to be looked at during the process of generating random peptide maps. Consider that one of these peptide maps is used to run simulation searches against the proteome *C. elegans*. The peptide map was generated using a complete database of proteomes, which encompasses approximately 433,880 proteins. The proteome *C. elegans* contains approximately 18,526 proteins. Therefore any peptide map generated would contain approximately 4% of *C. elegans* peptides. This amounts to a little more than one peptide per peptide map (of 30). This introduces a slight systematic error in results of *C. elegans* searches. The other two target proteomes are affected to a lesser extent due to their smaller proteome size. To correct for this problem several sets of peptide maps were generated. Each set for use against a particular target proteome was generated using all available proteomes except the target proteome itself.

Before the transcription process, the location of these 3 proteomes were located in 3 separate FASTA files:

E. coli: nr-enterobacteria.FASTA

S. cerevisiae: nr-fungi.FASTA

C. elegans: nr-otheranimals.FASTA

Certain ranges of numbers in the new 88-file system that contained the proteins of each of these proteomes were recorded. Since *C. elegans*, *E. coli*, and *S. cerevisiae* were mixed with other proteomes in larger files such as nr-enterobacteria (*E. coli*), it was necessary to exclude the content of these larger files as well in peptide map generation. This was not considered to be a problem.

Table 3.1: Database file omissions

Proteome	**Database File Omitted**	**Number of Proteins Available for Random Map Generation**
Escherichia coli	Nr-enterobacteria	415825 (96% of total)
Saccharomyces cerevisiae	Nr-fungi	411963 (95% of total)
Caenorhabditis elegans	Nr-other animals	391411 (90% of total)

3.3.2 Random Number Generators

To implement the process of assembling these random peptide maps a random number generator must be used. A number known as a "seed" is provided to a pseudo-random number generator as an initial integer to pass through the function. The pseudo-random number generator is a deterministic program that produces a completely predictable series of numbers (known as a 'stream'). A well-written PRNG creates a sequence that shares many of the same properties as a sequence of real random numbers. For example:

- The PRNG can generate any number in a range with equal probability.
- The PRNG can produce a stream with any statistical distribution.
- The stream of numbers produced by the PRNG lacks discernible patterns.

What PRNGs can't do is be unpredictable. Given the seed and the algorithm proper, the sequence of numbers can be predicted. On one hand, pseudo-random number generators have many useful applications. They work well for Monte Carlo simulations such as the ones being performed in this thesis. They also apply to other statistical sampling or simulation models. These applications typically require only moderate sophistication in their "randomness": patterns in a sequence don't resonate with naturally occurring sequences. There are ways of generating purely random numbers using special hardware. However it was decided that a PRNG was suitable for the task of generating random peptide maps.

The PRNG used for this thesis is the BSD degree-63 RNG, a popular PRNG developed in Berkeley that is used in many applications. This particular RNG uses the computer clock as the random seed value to generate pseudo-random sequence.

3.4 Search Parameters

The following sections detail the various parameters that were explored during the course of the project. Functionality for some parameters had to be programmed into the search engine (i.e. mass accuracy) while others were varied by altering the input data files (e.g. peptide map size). Each section contains a description of the parameters as well as how they were implemented. Results are listed in chapter 4.

3.4.1 Mass accuracy

Mass accuracy, Δm, is a descriptor of the mass confidence of input peptides, and is typically entered by the user when making a PMDS. Peptide matches are recorded provided they fit the following criterion:

$$m_{database} - \Delta m \leq m_{input} \leq m_{database} + \Delta m \qquad [3.1]$$

This parameter has a direct influence on the number of false matches returned in a database search.

Experimenters typically describe mass accuracy in either ppm or Da. The unit ppm, or parts per million, is a dimensionless value that can be converted to daltons by dividing by 1000000 and then multiplying by the associated peptide mass. Using uncertainties listed already in Da. indicates that no conversion is necessary. However, an uncertainty listed in ppm scales the mass accuracy with peptide mass. For example, consider two peptides, 800Da and 4500Da respectively. A mass accuracy of 50ppm translates to Dalton values as shown in the following table.

Table 3.2: Mass accuracy - *ppm* vs. *Da*.

Peptide	Mass	Mass accuracy
1	800	50/1000000*800=0.04Da
2	4500	50/1000000*4500=0.23 Da

Mass accuracy in ppm has an effect of scaling linearly with increasing peptide mass.

Many mass spectrometry experimenters prefer to use ppm since it reflects the propensity of peptide mass peaks to broaden with increasing peptide mass, leading to an increase in Δm.

Initially, mass accuracy was varied from 0ppm to 200ppm in 5ppm increments. Later, the range was reduced to 0ppm to 100ppm. Since most contemporary mass spectrometry instruments have accuracy of 100ppm or less, only a couple of simulations were made outside this range.

3.4.2 Number of Required Matching Peptides

This parameter designated how many matching peptides must occur between a RPM and a target protein for a match to be recorded. Typically users only have a handful of peptides, which they can be quite confident of. Question: What proportion of false matches can be screened out when the number of confident peptides is increased?

Simulations were performed where the number of required matching peptides was varied in increments from 1 to 10 for all target proteomes.

3.4.3 Random Peptide Map Size

One of the most important parameters influencing false match returns is the number of peptides contained in a peptide map. The number of proteolytic peptides used in database searches can vary substantially from experiment to experiment. The minimum number of peptides in a useful map is approximately 5.

Five sets of two hundred peptide maps were generated. Each set contained peptide maps of varying size, with the functionality to do this being programmed into the map generated described in section 4.4. The sizes chosen were 5, 10, 30, 50, and 80 peptides per map.

3.4.4 Missed Cleavages

Peptides are the result of enzymatic digestion of proteins, using high specificity enzymes, the most common being trypsin. However, in many cases the enzymatic digestion of a protein is incomplete. The number of missed cuts can be entered as a parameter in protein database searches in order to account for these "dual-peptides"
A database search engine matches experimental peptide masses to theoretical derived proteolytic peptide maps. To account for missed cleavage, the theoretical peptide maps can be expanded to include peptides where a cut has not been made. These new peptides can be matched to the experimental peptide masses, and in such a way account for the possibility of a missed cut. For example, for one missed cleavage, the following single peptide and derived dipeptides would be considered:

	Single Peptide	**Dipeptide**
Peptide List	TR	TRIQK
	IQK	
	IQK	IQKHMEQGDQSSSTTFNNGR
	HMEQGDQSSSTTFNNGR	

The highest number of missed cleavages can be entered as one of the search parameters.

During the simulations, missed cleavages were varied from 0-4. One can deduce immediately that the number of false matches will grow substantially with increasing number of missed cleavages. For instance, if the maximum number of missed cleavages entered is 1, all concatenations of two adjoining peptides are also added to the list of theoretical peptides under consideration.

3.4.5 Random Peptide Size

Smaller peptides are more likely to match with random database peptides. This parameter can affect the number of false matches being returned for a peptide maps that have constraints on the minimum peptide size.

To explore the effects of smaller and larger peptides, several peptide maps were generated. In each case, a particular limit was set on the smallest peptide. The values used were 5, 8, 10, and 12 amino acids.

3.4.6 Molecular Weight Range Constraint

The molecular weight of an intact protein can be used as a constraint on the candidate proteins. The object was to run one peptide map and vary the range of the intact molecular weight range.

3.5 Determination of Peptide Masses

As mentioned above, whole protein sequences are downloaded and cleaved using the enzyme trypsin. Trypsin cleaves at C-terminal sides of Lysine or Arginine. The cleavage is disallowed if the adjoining residue on the C-terminal side is Proline.

The amino acid residue masses used are listed in appendix B. The weight of a molecular peptide is:

$$m_{peptide} = \sum_{n=1}^{N} m_{residue} + 18.0153 \qquad [3.2]$$

18.0153 Daltons is the weight of a molecule of water, which is used to account for an N-terminal hydrogen and a C-terminal hydroxyl group of a peptide.

3.6 Simple 1 and 2 Peptide Searches

At this point, a complete proteome database is stored locally. Several random peptide maps have also been generated.

An engine for automating protein database search needed to be programmed. Several factors were considered during the programming stage. First, the database searches had to be executed with as little floating-point operations as possible, since processor power would limit the amount of possible research avenues. Second, the locally stored databases would have to be loaded into memory for efficient operation. Since the databases are quite large, programming steps would be taken to reduce as much as possible the size of random access memory needed.

3.6.1 One peptide searches

To start, a simple one-peptide matching engine was programmed. In this case, the target proteome *E. coli* was used. Both the random peptide map and the *E. coli* proteome were loaded into RAM. Subsequently, each separate peptide of the random map was used as input for a database search on *E. coli*. The resulting list of matches was stored and the process was repeated for the next peptide of the map. Once all peptides of the random map have been used, the next random map is loaded and the process is repeated. There are 200 random peptide maps used in each experiment. The algorithm in simple code is illustrated in figure (3.4).

The algorithm was scrutinized for proper operation. To start, a mass accuracy value of 0 was used. Since the peptides were 8 amino acids or larger in size the possibility of two peptides matching was statistically insignificant, thus no matches were expected, and 0 matches were returned. This was to check if there were any extraneous matches that would signify a malfunctioning algorithm. Subsequently, 50ppm was used. Result: Nearly every *E. coli* protein had at least one matching peptide.

A random sampling of the data was examined for veracity by performing protein database searches with the peptide map using the ProFound [11] search engine. The results were compared and non-matching results were investigated.

Most of the program debugging occurred at this stage. Invariably, small bugs were discovered during the course of the project, most did not required simulations to be repeated.

```
start
initialize variables
open peptide map
        ->calculate masses
        ->store in memory:
        ->index random map peptides
open E. coli proteome
        ->calculate masses
        ->store in memory:
        ->index random map peptides

for every random peptide map          // Loop 1
        for every E. coli protein     // Loop 2
                for every source peptide      // Loop 3
                        for every target peptide      // Loop 4
                                if      [(m random map peptide >= m database peptide - tolerance)
                                        AND (m random map peptide <= m database peptide +
                                        tolerance)]
                                record that a match occur between that random peptide
                                map and that E. coli protein then go on to next E. coli
                                protein (Loop 2)
                                end if
                        end for
                end for
        end for
end for
end
```

Figure 3.4: SimpleCode - One Peptide Search Algorithm

3.6.2 Two Peptide Searches

Next, the algorithm functionality was expanded to include two peptide searches. As before, the peptide map is used to search every target protein in *E. coli*. However, a match is recorded only if there are two non-redundant matching peptides. The algorithm, minus the initialization and data storing is listed in figure (3.5).

This is an algorithm with slightly more complexity than the one peptide case, and works perfectly fine for 2 peptides. However, this style of algorithm cannot be used in going to the n peptide case. This is because the above algorithm uses 4 for loops to allow 2 peptide matching, iterating twice through the source peptide list to find matches. To program a n peptide case a table of matches would have to be built, and redundant peptide matches screened out.

```
for every random peptide map          // Loop 1
    for every E. coli protein         // Loop 2
        for every source peptide      // Loop 3
            for every target peptide  // Loop 4
                if [(m_random map peptide >= m_database peptide - tolerance)
                    AND (m_random map peptide <= m_database peptide + tolerance)]
                record that a match occur between that random peptide
                map and that E. coli.
                end if
            end for
        end for
        if there was a match then execute following:
            for every other source peptide    // Loop 3
                for every other target peptide    // Loop 4
                    if [(m_random map peptide >= m_database peptide - tolerance)
                        AND (m_random map peptide <= m_database peptide + tolerance)]
                    record that a match occur between that random
                    peptide map and that E. coli protein then go on to
                    next E. coli protein (Loop 2)
                    end if
                end for
            end for
        end if
    end for
end for
end
```

Figure 3.5: SimpleCode - Two Peptide Search Algorithm

3.7 n Peptide Searches

Expanding the search engine capability to match n peptides was one of the major steps in the thesis. There were two major obstacles with programming for n peptide matches. First, matches would have to be non-redundant, as in two peptide searches, where the two peptides in a peptide pair match would not be used to match other peptides. Secondly, the algorithm had to be fast. As previously explained programming a new search engine from scratch was determined to be the easiest course of action rather than using the source code of an existing search engine and making extensive modifications.

For each particular target protein, all combinations of peptides are tested and the results recorded in a 2D matrix called *tfmatrix*. A TRUE is recorded if the two peptides match within the tolerance, if not a FALSE is recorded.

One of the programming challenges for 2+ matching peptides is ensuring non-redundant matches. Table (3.3) illustrates the problem. This is a simple programming problem when only two source peptides are involved.

Table 3.3: 2 peptide validity tables

	D1	D2	D3	D4
S1	F	T	F	F
S2	F	T	F	F

Invalid 2 peptide match
(2 SP matching one DP)

	D1	D2	D3	D4
S1	F	T	T	F
S2	F	F	F	F

Invalid 2 peptide match
(2 DP matching one SP)

	D1	D2	D3	D4
S1	F	T	T	F
S2	F	T	F	F

Valid Match
(non-redundant)

S* - Source peptide
D* - Destination peptide

T - Peptide Match
F – No peptide match

As can be seen in the table it is possible for 2 peptides on one database to match one peptide of the other, and since that is non physically possible it must be disallowed in the simulation.

The algorithm was tested for a number of real proteins and the results were compared with the results returned from using ProFound.

For the case of *n* matching peptides a more general algorithm was written that could count the number of non-redundant matches in a *mxn* tfmatrix (figure 3.6). The algorithm iterates column by column through the matrix, marking off (or crossing out) rows that fulfill a particular condition. Namely, if there is only one row with a TRUE in that particular column, mark it off and increment the count by one. If there is more than one TRUE, compare them. What we are comparing is the rightmost true. Mark off the row that has the lowest column index number for the last TRUE of the row. Consider the following matrix (table 3.4):

Table 3.4: n Peptide Validity Table

	D1	D2	D3	D4
S1	**T**	**F**	**F**	**T**
S2	**F**	**F**	**F**	**F**
S3	**F**	**T**	**F**	**F**
S4	**T**	**T**	**F**	**F**

While it is fairly easy to see visually that there are three non-redundant matches in this matrix, writing an algorithm to make the same determination, is as follows

1. Examines the first column. There are two TRUE's (row 1 and 4)

2. Select one row to mark off. The last TRUE of row 1 has column index 4. The last TRUE of row 4 has column index 2. Mark off the row with the lowest column index, i.e. row 4. Increment the count by one. (Table 3.5)

---

initialize variables
open peptide map
->calculate masses
->store in memory:
->index random map peptides
open *E. coli* proteome
->calculate masses
->store in memory:
->index random map peptides
for every random peptide map
for every *E. coli* protein
for every source peptide
for every target peptide
if [($m_{\text{random map peptide}} >= m_{\text{database peptide}}$ - tolerance) AND ($m_{\text{random map peptide}} <= m_{\text{database peptide}}$ + tolerance)]
record the peptide indices of the two matching peptides and increment a 'match' counter
end if
end for
end for
analyze tfmatrix and determine how many non-redundant peptide matches exist between this particular random peptide map and *E. coli* protein
end for
end for

Figure 3.6: SimpleCode - n Peptide Search Algorithm

Table 3.5: n peptide validity table - one peptide matched

	D1	D2	D3	D4
S1	T	F	F	T
S2	F	F	F	F
S3	F	T	F	F
~~S4~~	~~T~~	~~T~~	~~F~~	~~F~~

3. Repeat for columns 2 through 4.

4. 3 matches found in total.

Note that if the first row were marked off there would only be 2 non-redundant peptide matches.

The generalized code for tfmatrix sorting in listed in figure (3.7).

```
initialize:
tfmatrix[i][j] =0
taken[i]=F      i ranges from 0 to the number of destination peptides. Refers to which
                destination peptides have been matches previously to source peptides.
curmax=maxpepsize
The value of the rightmost true, originally set as a high value
max[j]=0        The index of the rightmost T value for each row.

for every COLUMN (i)
        take=-1         Best source candidate for a match, reset for each column.
        curmax=maxpepsize;
        for every ROW (j)
                if (current cell ==TRUE ) AND  (max[j] < curmax) AND (taken[j] ==
                FALSE))
                        take = j;
                        curmax = max[j];
                end if
        end for
        if (take != -1)  {
                taken[take] = TRUE;
                matchingpeps++;
                end if
 end for
if (matchingpeps >= numpeps)
        nummatches++;
        record the match
end if
```

Figure 3.7: SimpleCode - tfmatrix operation

3.8 Cluster Computing

A cluster system was developed to help increase throughput. A remote TCP/IP interface was programmed into the pepstat client. Pepstat clients were then placed on available Windows 32 bit systems. A separate 'control' program was programmed to query the clients. The top frame of figure (3.8) illustrates communication between the control and the pepstat clients. The bottom frame shows a typical case of the functioning system. Pepstat 1 is transferring results to the control, which will issue new commands after the transfer is complete. Pepstat 2 is processing a task. Pepstat 3 had completed a task and is waiting to be queried by the control before transferring data.

Parameter ranges are input in into the control program, which then generates a task list. For instance, the following is one task

> PEPMAP1 against *C. elegans* - MC: 2 / MP: 4 / U:35 / MINAA 8 /

If the control is resuming a previous run, a subset of tasks are pre-marked as complete.

The control then iterates through the pepstat client list (each pepstat client IP is input beforehand). If control finds a client that is inactive, if so, it allocates a task and marks the task as 'allocated but incomplete'. It then allocates tasks to all other pepstat clients in this manner until a task has been allocated to all inactive pepstat clients.

When the control contacts a client, it establishes a TCP/IP connection. It then sends a command 'packet', consisting of a query or a task it wants to allocate to the client. The client responds 'OK' if it is free and has started the task, or 'BUSY' if it is already processing a task, in which case the control would move on to the next client.

Once a command packet is received, the remote interface of a client sets the appropriate parameters in pepstat's GUI, and then runs the search by running the method that would normally be run if a user hit 'Go' in the pepstat window.

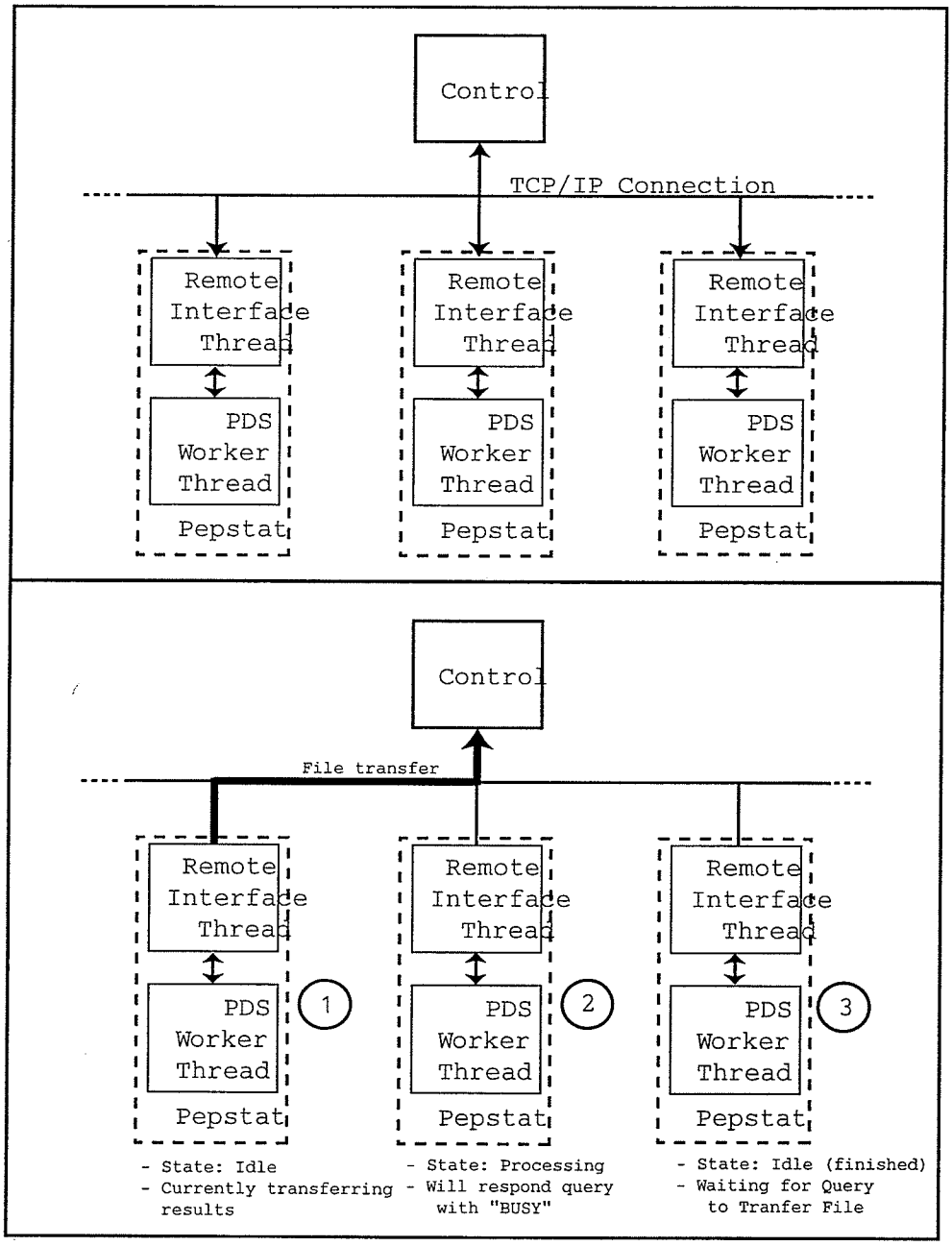

Figure 3.8: Pepstat and control functioning.

The pepstat TCP/IP interface was added fairly effortlessly. The most difficult part of programming the interface was enabling it to run inside a thread. The only socket class available in VC++, CSocket, could not be created outside of the worker thread that intended to use it, and this created some logistical problems with the programming. It was required to create a Socket outside of the thread, detach and sent the handle into the thread, create a CSocket class in the thread, and re-attach the handle.

There were certain limitations regarding which computers could be used in the cluster. These requirements were

1. >= 500MHz processor
2. min. 256 Megs. RAM
3. Win32 operating system
4. Permanent Internet connection

The pepstat program was designed to be minimized into the system tray, as well as running on the lowest thread priority, _IDLE, the program will only run on idle cycles, ensuring that the program does not interfere with the normal duties of the computer in question, in fact all the computers used in the project were non-dedicated computers. Limitation 1 was strictly a convenience limitation, as the program would run on any speed computer, but anything slower than a 500MHz computer ran pepstat too slowly to be of much use. Limitation 2 was necessary. When pepstat is run, two large sets of data need to be loaded into RAM, namely the RPM and the target proteome. With proteomes such as *C. elegans*, this data could easily occupy 100 Megs of RAM. 256 Megs are needed to ensure there is an adequate supply of RAM for other applications. Limitation 3 occurs due to the non-portability of the program to other OS systems. Limitation 4 is necessary to allow the control program to continuously query the pepstat clients.

All cluster computers used were resident computers of the time-of-flight laboratory[12], with the exception of one computer that was located at the author's address. 4 computers were used (table 3.6), with some other computers added temporarily.

Table 3.6: List of machines used in a cluster configuration

Computer	Processor	RAM
Development Machine	500 Mhz Celeron	768
Lab Machine 1	1 GHz Pentium III	1047
Lab Machine 2	800 MHz Pentium II	527
Home Machine	1 GHz Athlon	256

3.9 Development Platform

All code written for the thesis was programmed using Microsoft Visual C++, version 6.0. A PC platform was used, driven by an Intel Celeron (500MHz) processor, 384 Megabytes of RAM, running Windows 98 SE2.

[12] Time of Flight Laboratory, Room 506 Allen Building, Department of Physics, University of Manitoba, R3T 2N1

Chapter 4

Results

This chapter details the results obtained by varying a single search parameter while holding all others constant. Each section explains the specifics of particular search parameters, how it is used in a protein database search, followed by simulation results.

4.1 Standard deviation in several similar simulations

For the most part, the data provided in this chapter and the next are the results of simulations of one or more random peptides maps against *E. coli*, *S. cerevisiae*, and *C. elegans*. Each simulation generates a new data point in parameter space.

This section is an endeavor to consider the error involved in these simulations. In this case we are interested in calculating how accurate each data point is, keeping in mind that each data point is a mean value derived from 200 peptide maps. Hence, we are interested in the standard deviation from the mean.[13]

If one PDS is repeated with the same parameters and different random peptide maps, a certain value of standard deviation from the mean can be calculated. In this case we are interested in standard deviation from the mean, which is based on the central limit

---

[13] For the user of a PDS engine, simple standard deviation would be an important factor in quantifying results. Calculating over one set of 200 peptide maps, the standard deviation is quite large and hence it is hard to validate patterns in the data. However, for the programmers behind the search engine, standard deviation for the mean is a more important indicator of underlying trends.

theorem. It states that given a distribution with mean μ and standard deviation σ^2, the sampling distribution of the mean approaches a normal distribution with a mean (μ) and variance σ^2/N, where N is the sample size used to generate each mean. Thus

$$\sigma_m = \frac{\sigma}{\sqrt{N}} \qquad [4.1]$$

σ_m - standard deviation from the mean over several simulations

σ - standard deviation of one set of peptide maps

N - number of samples from the population distribution(e.g. 200)

σ is calculated over one set of peptide maps. In this example, N=200.

If one only measurement is made, there is much larger error since there is no longer a division by 14.14. This is the standard deviation experienced by someone running a protein database search. However the standard deviation from the mean is used here to examine characteristics that are statistically significant and that PDSE programmers might be interested in. However the simple standard deviation will be calculated for the example that follows.

5 simulations were performed while varying the random peptide maps (the only random element in the simulations), and keeping all other variables constant. The simulations were performed over the three proteomes. Thus we are examining three points in parameter space. One can infer that the standard deviation from the mean for all other simulations would be similar to this case.

5 separate peptide map sets were created, possessing the same characteristics: 8 minimum amino acids, 30 peptides per map, and 200 peptide maps per set. These 5

peptide maps were run against *E. coli*, *S. cerevisiae*, and *C. elegans*, while holding the following parameters constant:

- Mass accuracy: 50ppm
- Minimum matching peptides: 2
- Missed Cleavages:0
- Molecular Weight Range: Infinite
- Monoisotopic Masses

The results are summarized in table (4.1):

Table 4.1: Results - standard deviation of 5 peptide maps

	False Matches		
	E. coli	*S. cerevisiae*	*C. elegans*
Peptide Map Set 1	33.08	115.15	275.48
Peptide Map Set 2	33.435	112.72	270.505
Peptide Map Set 3	32.42	113.75	270.85
Peptide Map Set 4	33.07	113.475	272.96
Peptide Map Set 5	31.865	112.56	270.17
Mean	32.74	113.53	271.99
σ	10.01	28.56	66.74
σ_m	0.63	1.03	2.23
σ_m (CLT)	0.708	2.02	4.72

σ – standard deviation for an individual measurement
σ_m – standard deviation from the mean calculated from the 5 peptide map sets.
σ_m (CLT) – standard deviation from the mean calculated using σ and CLT

The last row of the chart is the standard deviation from the mean calculated using the central limit theorem. The standard deviation used for the calculation is the standard deviation from the mean, which is a close approximation to the standard deviation of the parent population.

Actual values of σ_m are less than predicted values using the CLT. For large values of n(number of simulations), there should be no difference between the values. Thus, to calculate errors for all future measurement we can use the central limit theorem to calculate σ_m. In fact the CLT clearly applies to this situation and eliminates the need to perform several simulations as part of an error analysis. All subsequent error reported in this section use error values calculated from the CLT.

4.2 Mass accuracy

4.2.1 Simple Case

The plot shown in figure (4.1) illustrates the dependence of false matches on mass accuracy. Simulations were performed using (PM) on the *E. coli* proteome. Variables held constant were

Missed Cleavages: 0

Error type: Monoisotopic

Number of Peptides per Map: 30

Minimum # of amino acids in a peptide: 8

Minimum Number of Required Matching Peptides: 2

The mass tolerance was varied from 0 to 100ppm in 5ppm increments. Note that the value of FMPPM only becomes 0 when MA is 0. Such a mass accuracy is, as yet, unattainable, so in order to eliminate false matches, other parameters would need to be further restricted.

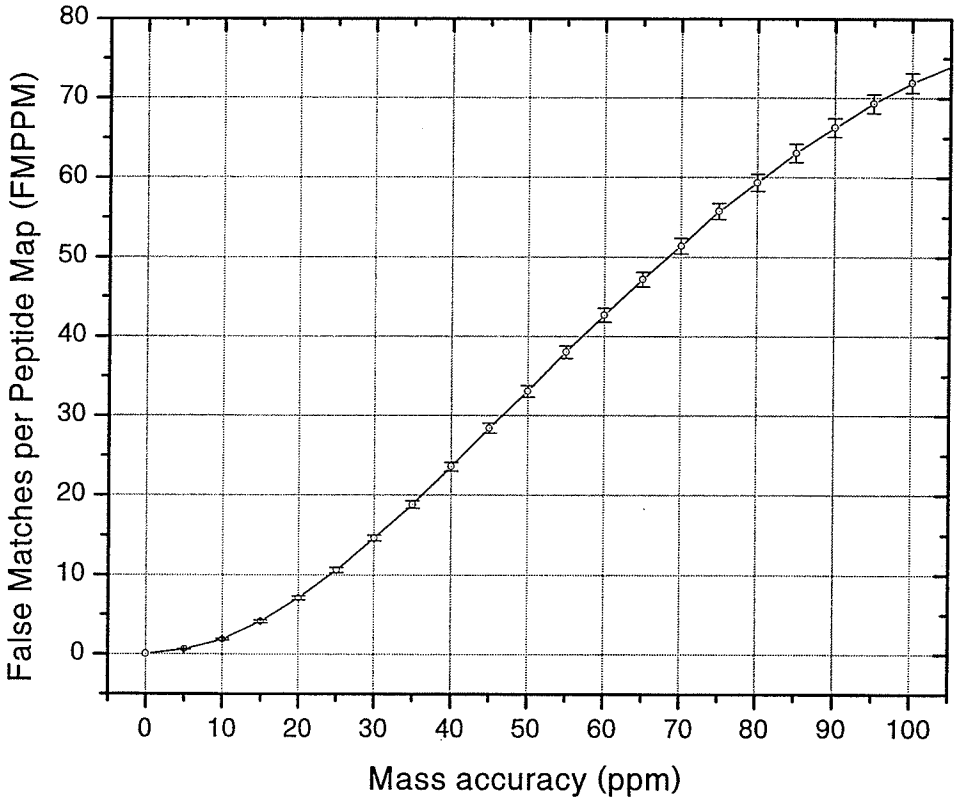

Figure 4.1: Results - false matches per peptide map vs. Mass accuracy for *E. coli*

Looking at the figure one can see two traits that are prevalent in all following mass accuracy dependant graphs. First, the graph is smooth, with no abrupt characteristics in isolated mass ranges.

Second, the shape of the curve is sigmoid, increasing only slowly at first, reaching a maximum increase rate, followed by a plateau. A few more points were taken for large

mass accuracy values, to examine the behavior of the plot. The expanded curve is shown in figure (4.2):

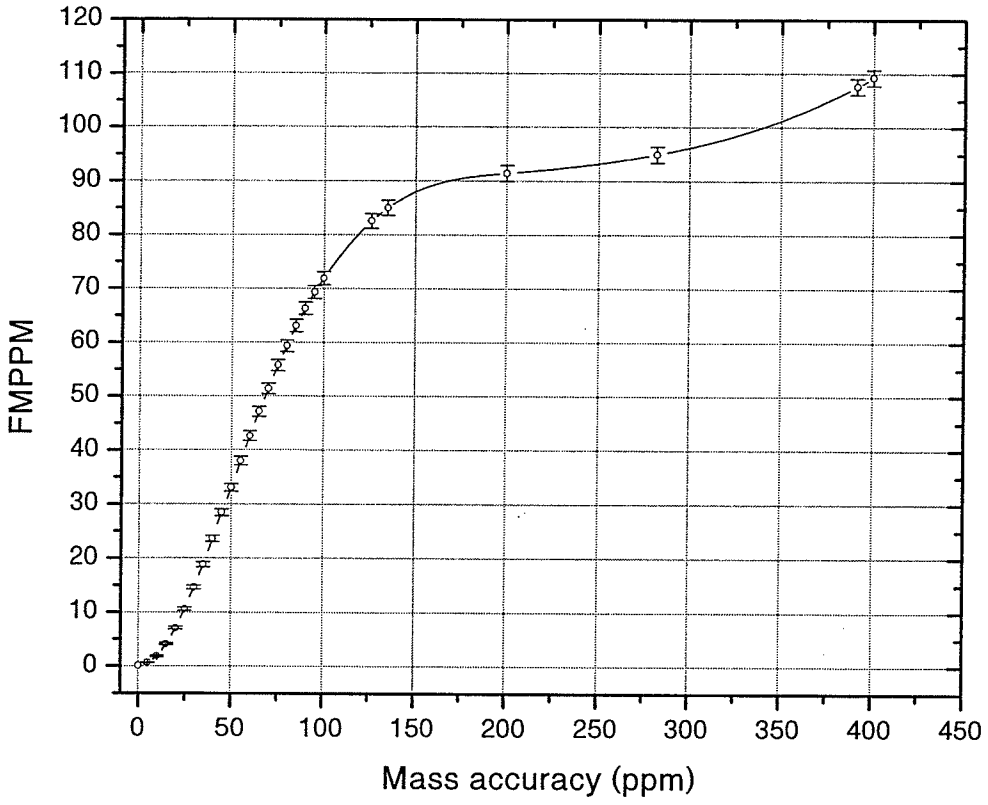

Figure 4.2: Results - Wide Field FMPPM vs. MA for *E. coli*

The ppm values used were 0-100 in 5 ppm increments, as well as the following ppm values: 126, 135, 200, 282, 391, 400.

After flattening out, the curve once again starts a new rise at approximately 350ppm.

The sigmoid shape of the plot can be explained by looking at the characteristics of the peptides that comprise both the peptide maps and the target proteomes. Figure (4.3) illustrates an interesting characteristic of these peptide masses. This plot is a histogram of peptide count vs. mass. Peptides from the *E. coli* proteome were binned and counted. The procedure was repeated for peptides contained in random peptide number 1. The bins are 0.1 Daltons wide. The plot shown is for an illustrative range. The behavior is

uniform throughout the range of peptide masses. The red histogram columns indicate *E. coli* peptides, while green columns denote random map peptides.

It can be immediately noticed that peptide masses fall into 'groups', with peptide mass gaps, where there are no counts:

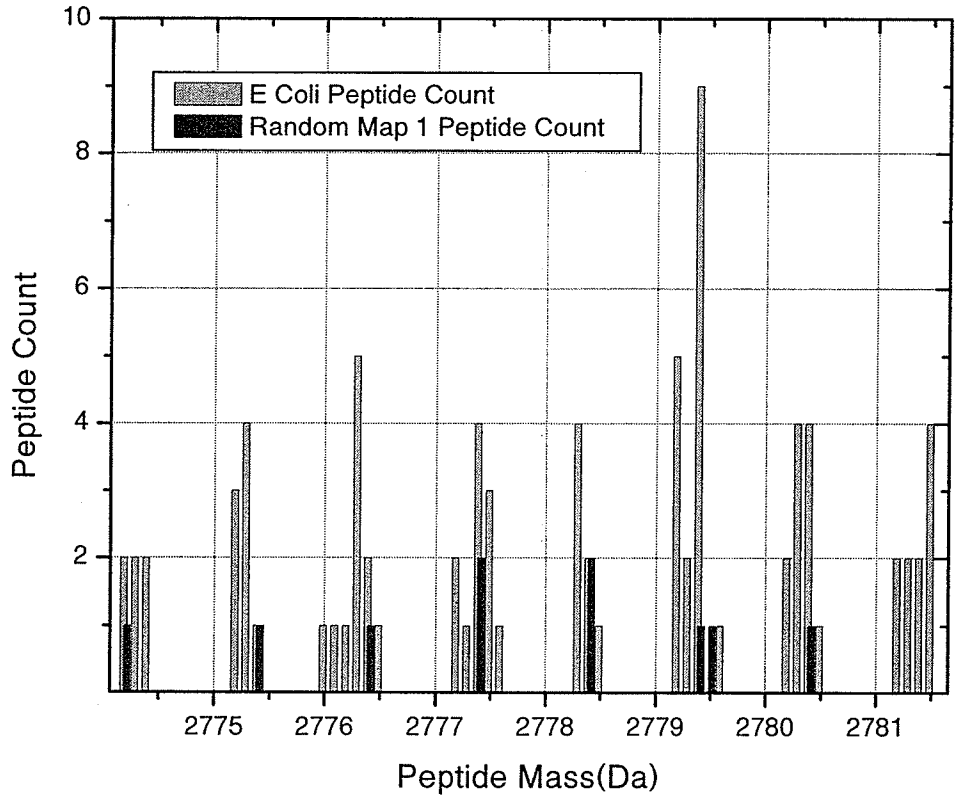

Figure 4.3: Mass distribution of Peptide Map and target proteome peptides.[17]

At low mass uncertainty values there are few if any matches. As the mass uncertainty is increased there is a large increase in overlap between the green and red columns of figure (4.3), which leads to more false match returns. There is a point of saturation however, where green and red groups completely overlap. Subsequently, there are little other peptides to match until the mass accuracy has grown so large that peptides contained in a particular green group n start matching red peptides in red groups (n-1) and (n+1). At this

point there is another steep increase in the curve, which is what the data indicates around the 350ppm range.

4.2.2 FMPPM vs. mass accuracy compared for *E. coli*, *S. cerevisiae*, *C. elegans*

Similar simulations were performed against the 3 proteomes. The ppm values used are the same as in 5.2.1, the *E. coli* case.

The plots in fig. (4.4) and fig (4.5) show the results. The top two charts show the data without normalization. The overall shape of each plot is similar to the one for *E. coli*, and not surprisingly, *C. elegans* has the most FM's over the whole range, followed by *S. cerevisiae*, and finally *E. coli*.

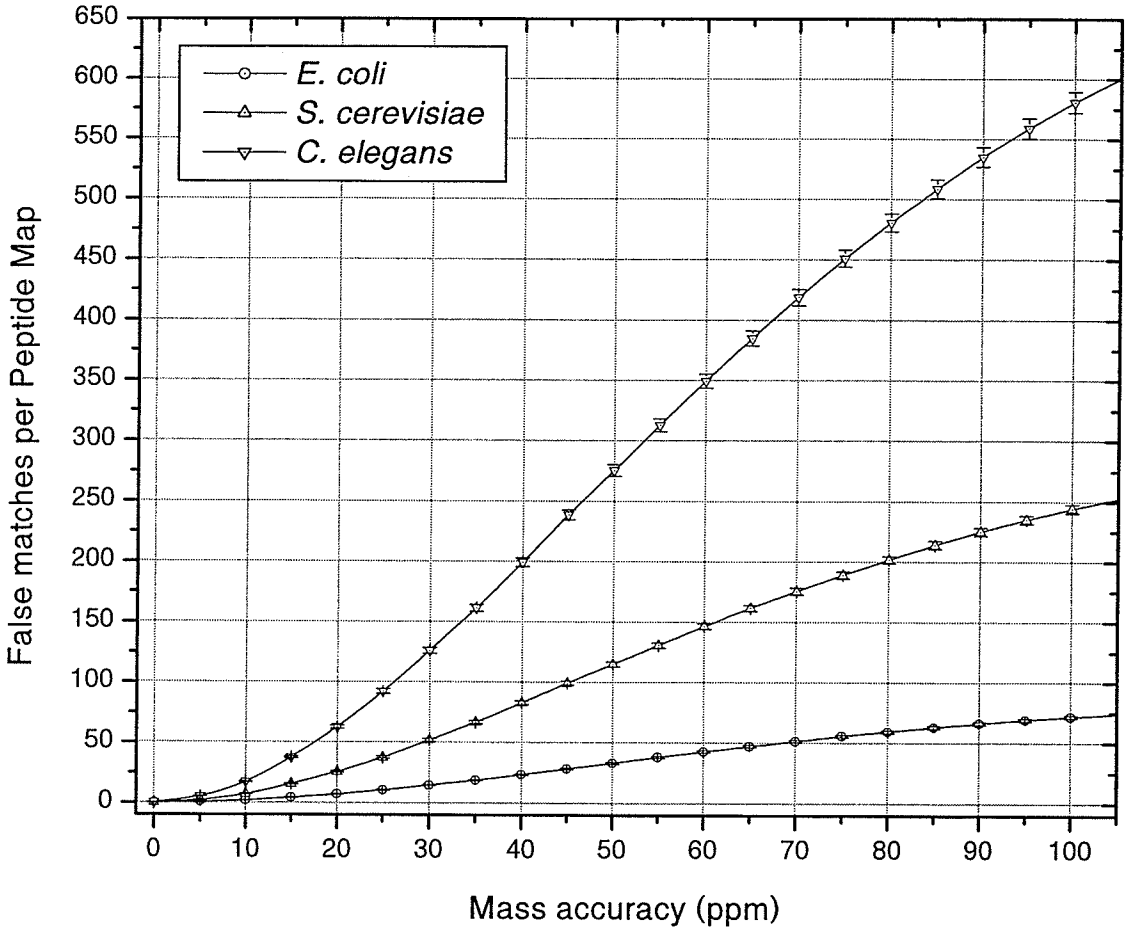

Figure 4.4: FMPPM vs. MA for 3 proteomes (no normalization)

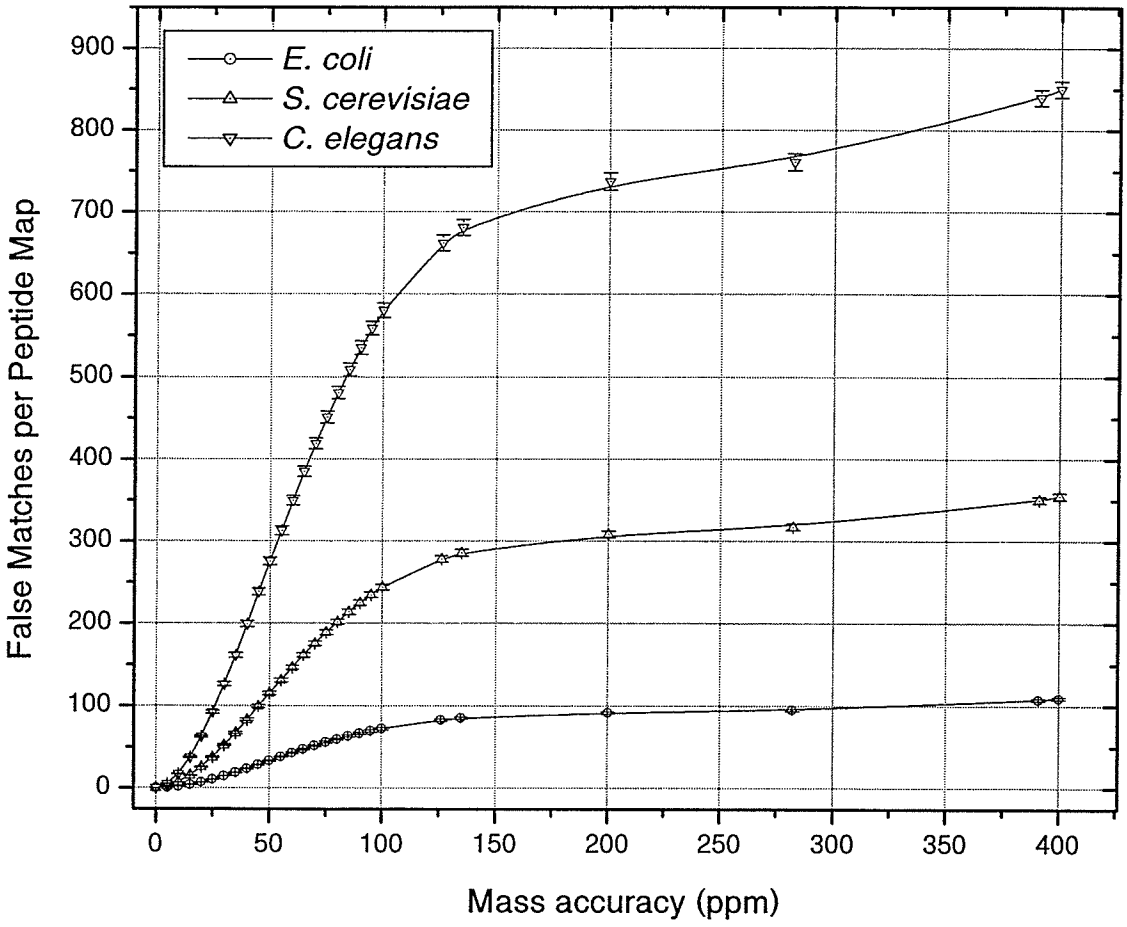

Figure 4.5: Wide-field FMPPM vs. MA for 3 proteomes (no normalization)

It is interesting to analyze the 3 plots after compensating (normalizing) for different proteome sizes. For instance, C .elegans has many more proteins, and therefore any search with a RPM will yield substantially more results than for *E. coli*. One can normalize the results, however, to take into account the relative size of the proteomes. In this case, a handy normalizing size is that of *E. coli*, the smallest of the 3 proteomes. Since the three target proteomes have varying sizes, false match returns had to be normalized. The adjusted number of false matches was normalized to number of peptides contained in the proteome *E. coli*.

$$FM_{nor,\Delta m} = FM_{\Delta m} * \left(\frac{132486}{pepcount} \right) \qquad [4.2]$$

the term in brackets is the ratio of peptides in the *E. coli* proteome to peptides in the proteome in question.

Charts 4.6 and 4.7 display the normalized results. For any particular ppm value, one observes interesting behaviour. Specifically, the *E. coli* plot, in black, shows the fewest FM counts, while *S. cerevisiae* shows the largest, with *C. elegans* in the middle. This difference is striking considering that at low ppm values, the FM returns of *S. cerevisiae* is noticeably greater than that of *E. coli*. Using this information, one can conclude that false matches vary depending on the particular target proteome, and not simply because of varying sizes. This result is useful to a PDSE programmer, who may use these results to add an offset to a scoring algorithm.

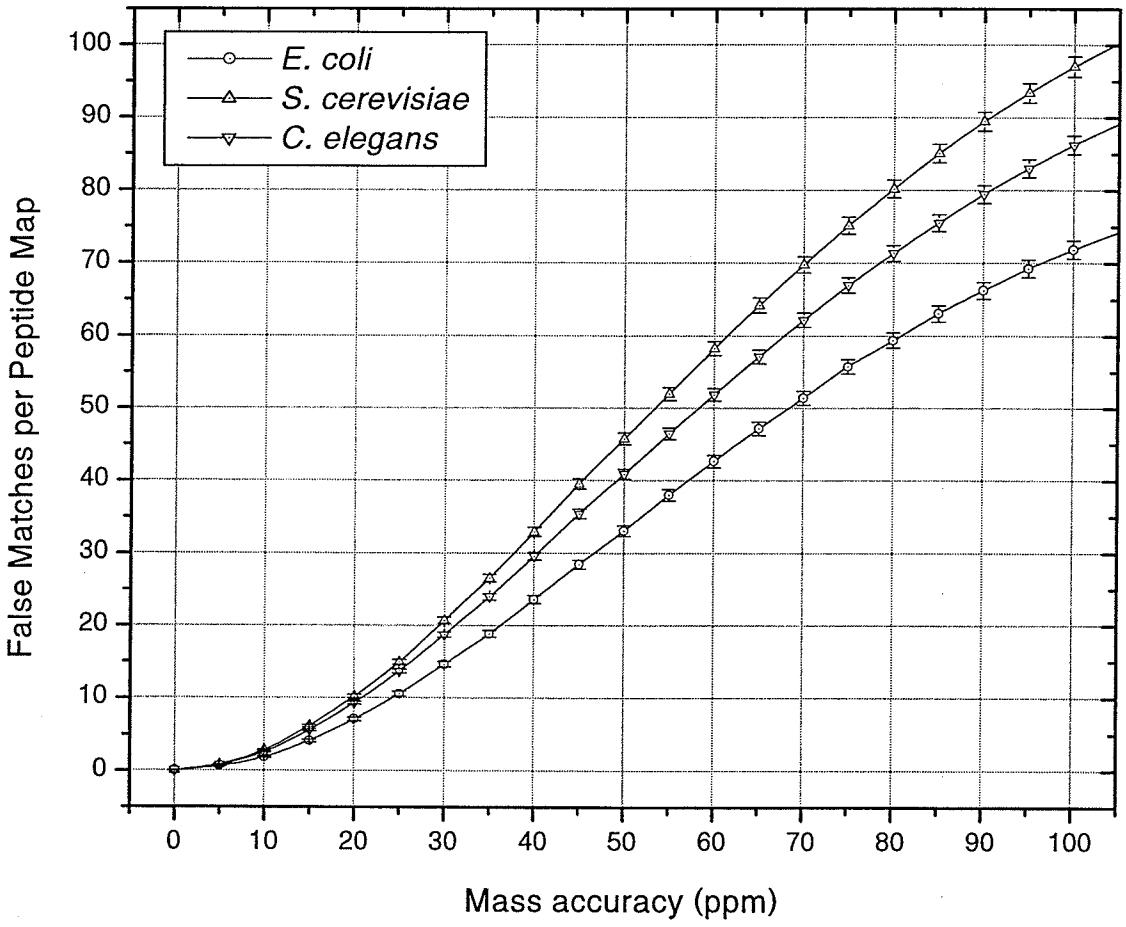

Figure 4.6: Normalized FMPPM vs. MA for 3 proteomes

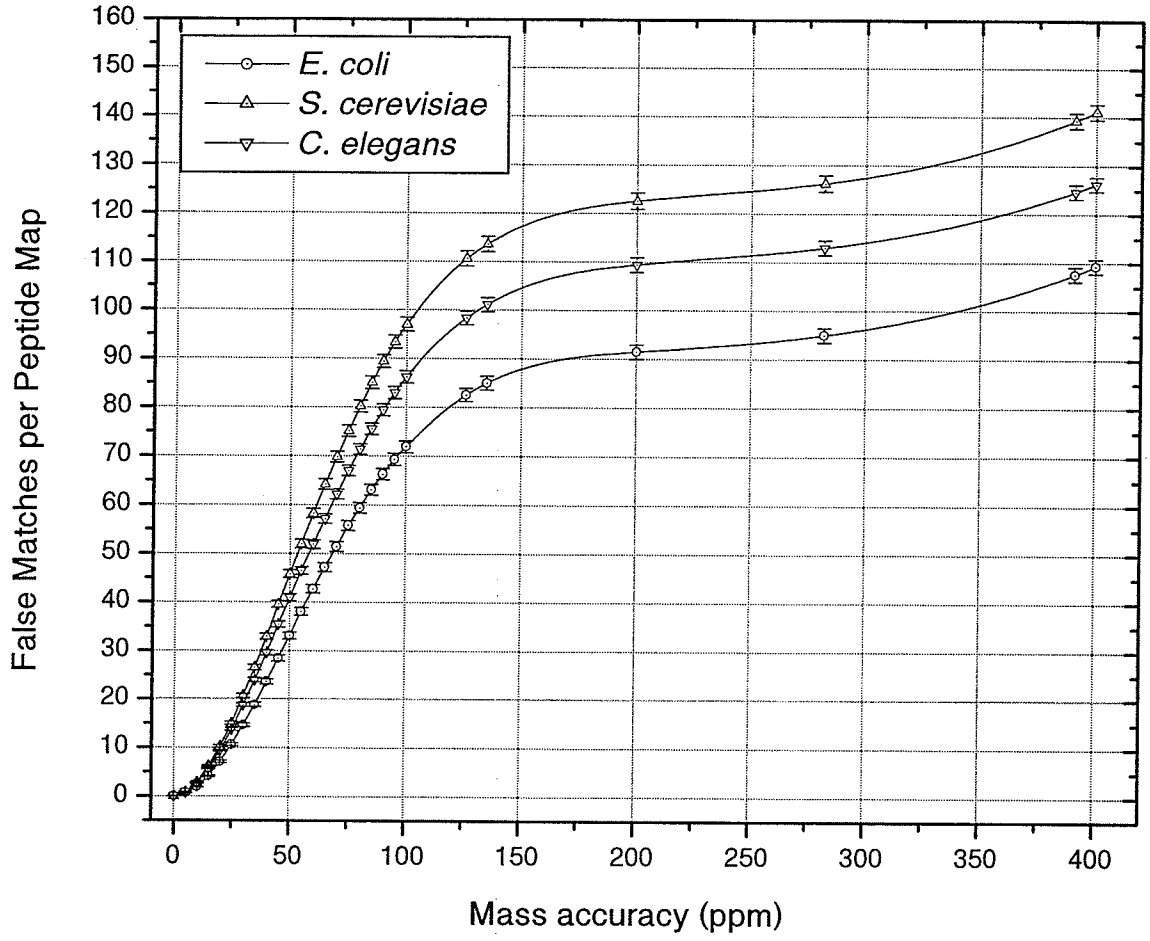

Figure 4.7: Wide-field normalized FMPPM vs. MA for 3 proteomes

Table (4.2) displays the total number of peptides for each proteome as well as the average size of a peptide.

Table 4.2: Proteome and peptide map statistical information

	Number of proteins	**Peptide count**	**Av. peptide AA count**
E. coli	4286	132486	10.24
C. elegans	18526	892260	8.81
S. cerevisiae	6217	333216	8.87
Random Peptide Map	200	6000	18.68

Note that the minimum amino acids of the random peptide map are 18.68. One would expect that since *E. coli* and the random peptide map are most similar in terms of average peptide size, the corresponding false matches would be the highest. However, figure (4.6) shows just the opposite.

Figure (4.8) shows the peptides from each proteome map as well as the RPM, sorted by mass into bins 25 Daltons wide. The counts from each proteome were normalized to the number of peptides in the random peptides map (an arbitrary value), using a variant of equation [4.2]

$$normalizedcounts = \left(\frac{6000}{pepcount} \right) * counts \qquad [4.3]$$

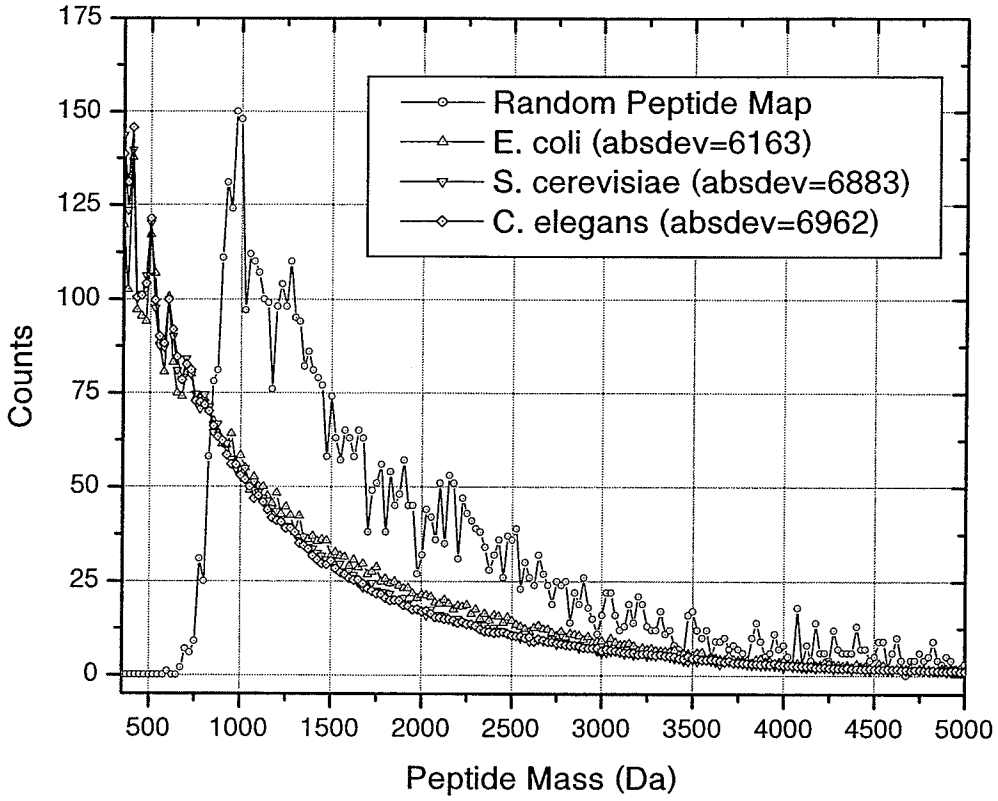

Figure 4.8: Proteome mass distribution

For each proteome, absdev is calculated using the following formula:

$$absdev = \sum_{i=1}^{1000} |counts - RPMcounts| \qquad [4.4]$$

where i is the bin number. The value of absdev for each plot gives an idea how well each plot matches the random peptide map plot. As expected from table (4.2), *E. coli* is the closest match to the RPM distribution. As well, note that the average peptide size for *S. cerevisiae* and *C. elegans* are almost equal, yet they have quite dissimilar FM returns.

One can conclude from this data that each proteome has an inherent quality that affects the quantity of false matches. This may be related to homology - a common

relation or ancestry between organisms. One important point, however, is that this can be represented as an absolute value, since the random peptide map consists of peptides from all other organisms. It is then possible to generate an absolute 'homology' value for each proteome and incorporate this data in protein database scoring algorithms. This can be useful when running a PDS where the experimenter is not sure of the source organism. Since the matches can quite possibly be from multiple organisms a refinement of the scoring algorithm using the aforementioned 'homology' value might be quite useful.

4.3 Number of Required Matching Peptides

Next, simulations were performed to observe the effects of varying the number of required matching peptides. The graph in figure (4.9) shows the results for each proteome.

Parameters held constant were:

Mass accuracy: 50ppm

Missed Cleavages: 0

Molecular Weight Range: Infinite

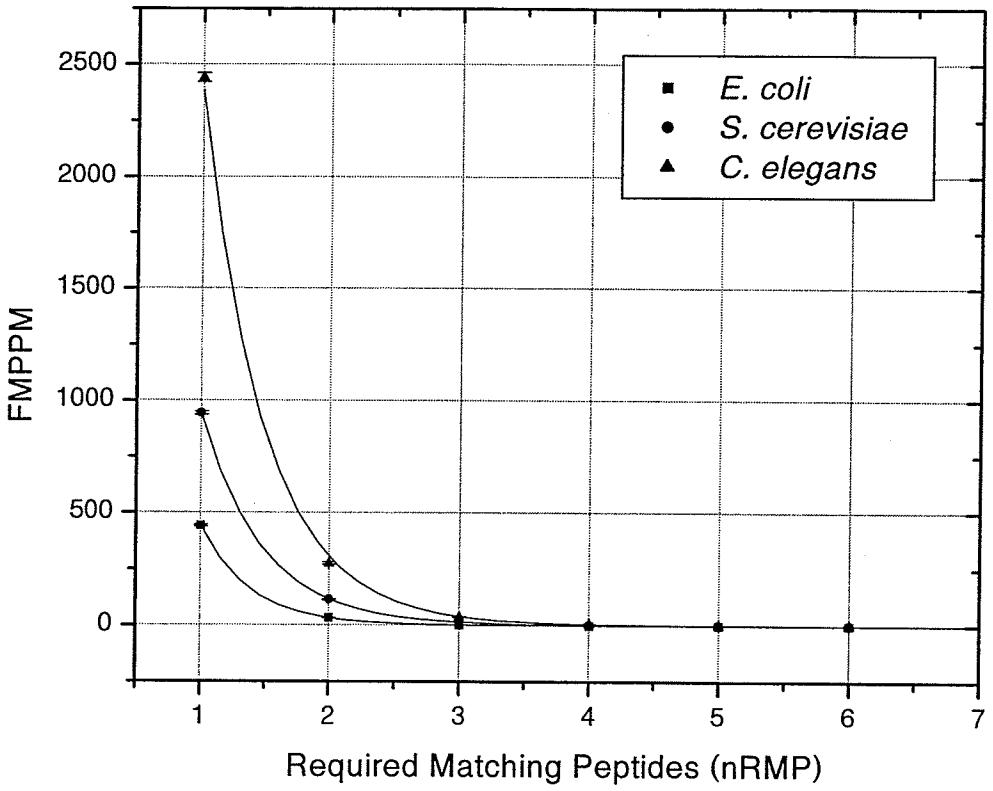

Figure 4.9: FMPPM vs. nRMP

The results show a striking difference in matches based on the x parameter. It was noticed that the data approximated a decreasing exponential so fit was attempted on the data. The equation that was fitted was

$$y = y_o + Ae^{-(x-x_0)/t_t} \qquad [4.5]$$

The chi-square values in each case are listed in table 4.3.

Table 4.3: X^2 values for exp. dec. fit of FMPPM vs. nRMP

Proteome	**X^2 value**
E. coli	0.564
S. cerevisiae	0.37
C. elegans	36.89

in this case the first two fits are excellent, while the third is fair.

For each increasing value of nRMP of one, there is a 10-fold drop in the number of matches. In the case of *E. coli*, a value of 4 nRMP is sufficient to effectively eliminate false matches. *S. cerevisiae* needs 5, and *C. elegans* needs the most, with 6. This trend is not wholly unexpected, due to the size of each respective proteome. Table (4.4) illustrates required nRMP and MA values required to drop false matches below 1.

Table 4.4: nRMP values required to reduce false matches below 1

nRMP Value	***E. coli***	***S. cerevisiae***	***C. elegans***
1	441.825	942.9	N/A
2	33.08	115.15	275.48
3	2.27	14.3	36.57
4	**0.17**	1.84	6.77
5	0.025	**0.19**	1.46
6	0	0.035	**0.375**
10	0	0	0

These results demonstrate that one particularly effective way of gaining confidence in matching is isolating the near-exact mass of a handful of peptides, as few as 4 peptides.

It is possible to combine these observed effects of nRMP with the observed effects of MA. Figure (4.10), illustrates the effects of this parameters on plots of FMPPM vs. mass accuracy. There is an approximate10 fold difference between the false matches of each plot. Each graph reaches 0 at different mass accuracy values. Table (4.5) illustrates the ppm value at which the value of FMPPM is 0.

Table 4.5: Required nRMP and MA values to achieve 0 FMPPM.

nRMP Value	**Mass accuracy needed**
1	N/A (7.505 FMPPM @ 0ppm)
2	N/A (0.015 FMPPM @ 0ppm)
3	5
4	20
5	35
6	75
10	> 100

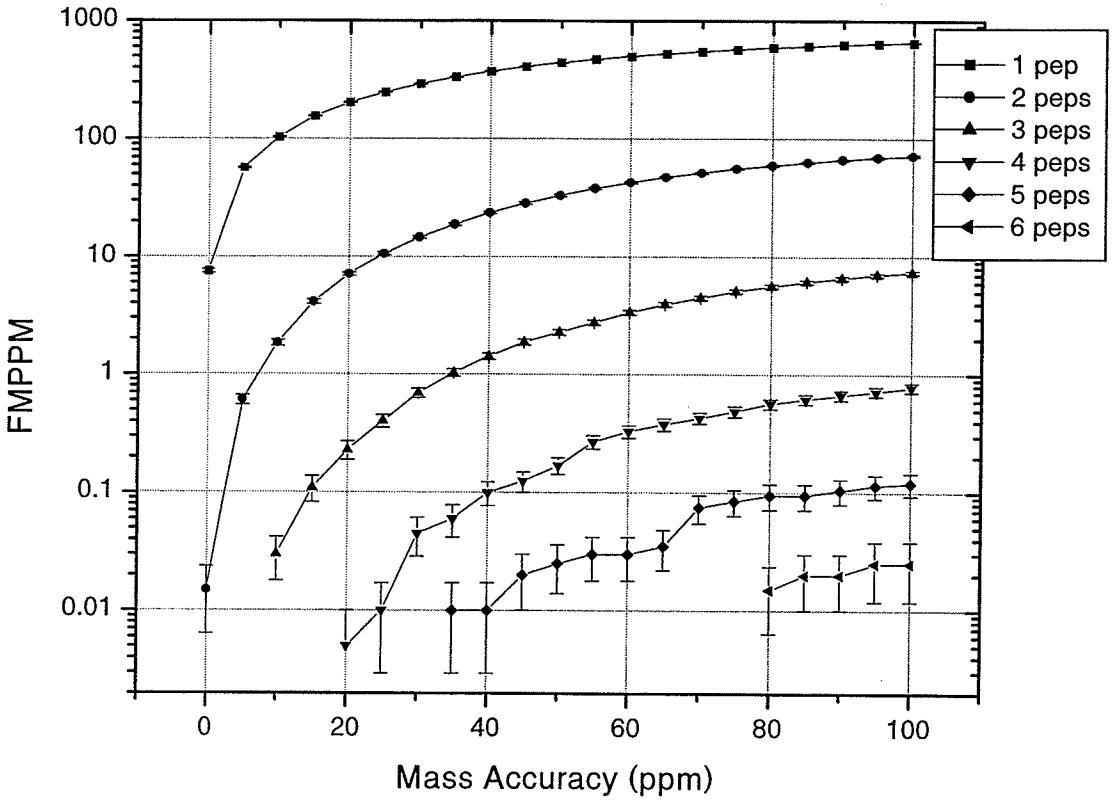

Figure 4.10: FMPPM vs. MA and n(log plot)

4.4 Random peptide map size

Fig. (4.11) shows the results of performing simulations with the five peptide map sizes (5,10,30,50,80). The simulations were repeating while varying the number of required matching peptides from 2 to 4.

Small peptide maps are much less prone to false matching than larger ones; in fact, the 80 peptide map is has approximately 50 times the number of false matches that the 10 peptide map (for 2 nRMP).

Increasing nRMP to 3 or so predictably decreases the FMPPM (as discussed in 4.4), however the rate at which the value increases from 30p to 80p maps for each line is significantly different (table 4.6):

Table 4.6: Ratio of FMPPM between 80p and 30p maps

RMP	**Ratio of FMPPM(80p)/FMPPM(30p)**
2	5.52
3	13.68
4	32.35

Figure 4.11: FMPPM vs. peptide map size for 2,3,4 nRMP

4.5 Missed Cleavages

It is seen from the results in figure (4.12) that incomplete enzymatic cleavages yield higher values of FMPPM. The relationship is fairly linear over the three plots. The relative change of FMPPM is most important for few missed cleavages. Going from 0 to 1 MC, the hits approximately double, with lesser relative increases later on. One missed cleavage effectively doubles the number of theoretical peptides under consideration, which is enough to put most of the experimental peptide maps over 3 matching peptides. Each extra missed cleavage essentially adds one more 'layer' of available theoretical peptides, as shown in the results.

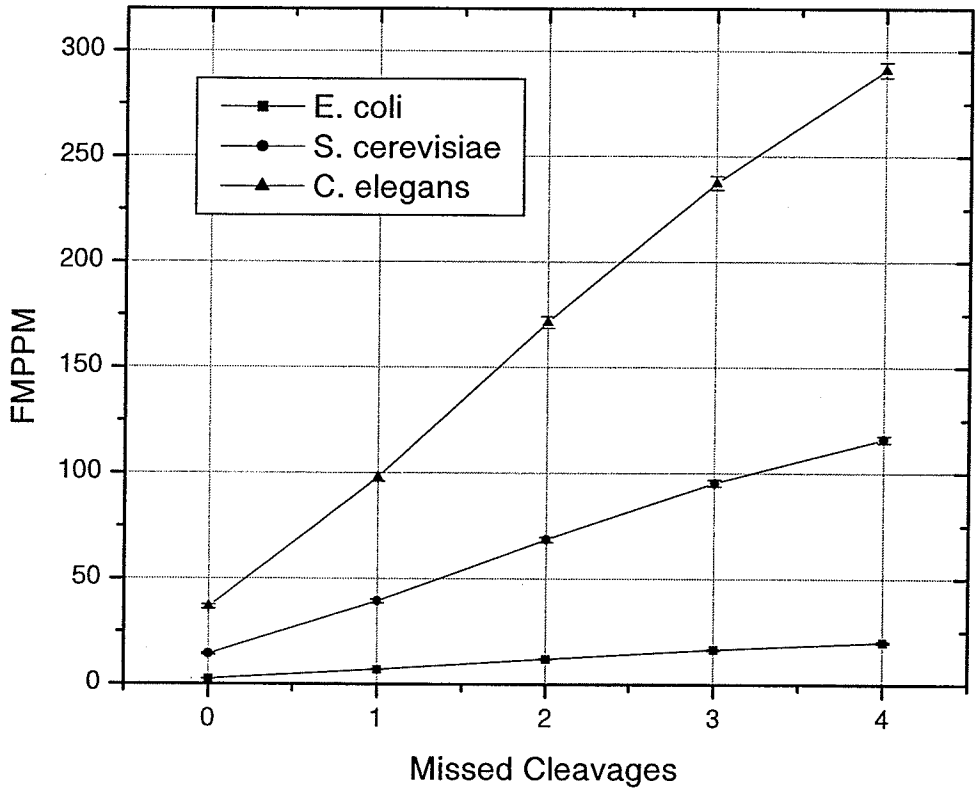

Figure 4.12: FMPPM vs. missed cleavages

In cases with mixed parameters the overall pattern of FMPPM vs. missed cleavages appears unchanged; figure (4.13) illustrates this dependence for 4 separate configurations of 2 other parameters, minimum number of matching peptides and mass accuracy. Each plot resembles other differences existing between the 'baseline of each plot'. The two lower plots illustrate the similarity between MP and MA for generating similar plots. In this case, the plot is almost identical when MP is increased from 3 to 5 with an offsetting increase in MA from 10ppm to 50ppm. As before, the increase in random matches is only about two-fold for an increase in missed cleavages of 1. This is a relatively small increase compared to other more sensitive parameters

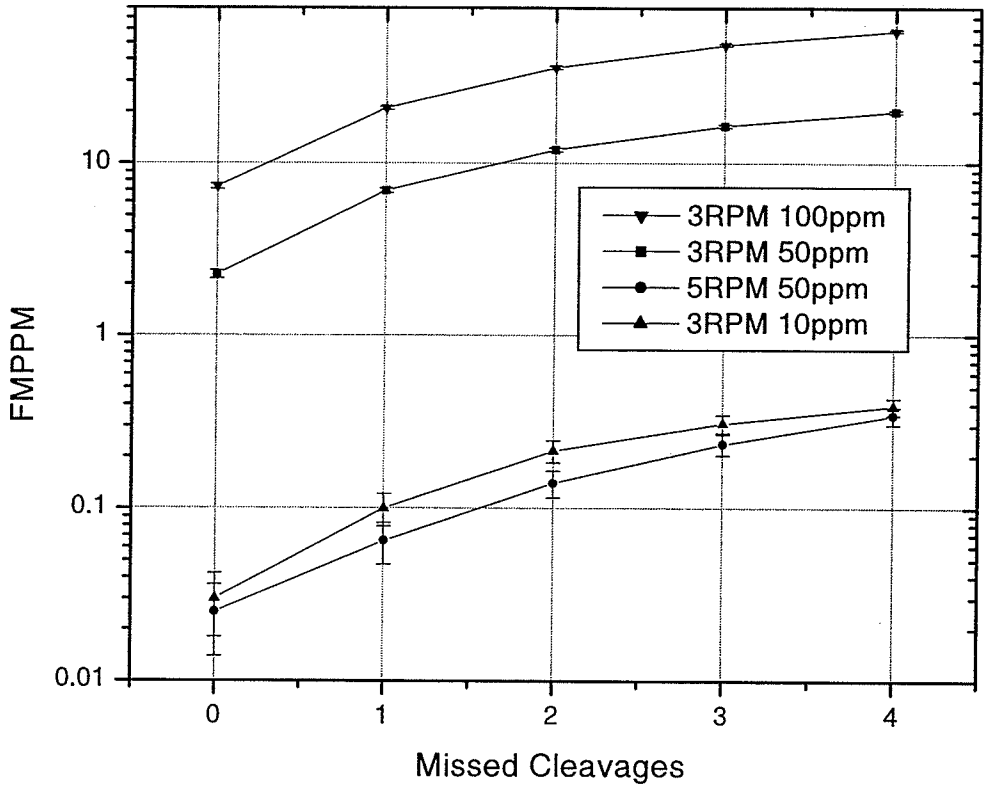

Figure 4.13: FMPPM vs. combinations of nRMP, MC, and MA parameters

4.6 Minimum peptide Size

The size of experimental peptides used in a search is important. A glance at figure (4.8) indicates that the majority of peptides in any proteome are in the low mass region. One would infer that increasing the size of peptides would have a logarithmic effect in minimizing false matches.

Twelve simulations in total were run to explore this effect. 4-peptide maps were generated that had a limit on the smallest peptide. These limits were 5, 8, 10, and 12. Maps with 10, 30, and 50 peptides were generated for each of the lower limits.

The results in fig (4.14) clearly illustrates that logarithmic decreases in false matches is indeed what happens, and this plot shape is independent of peptide map size.

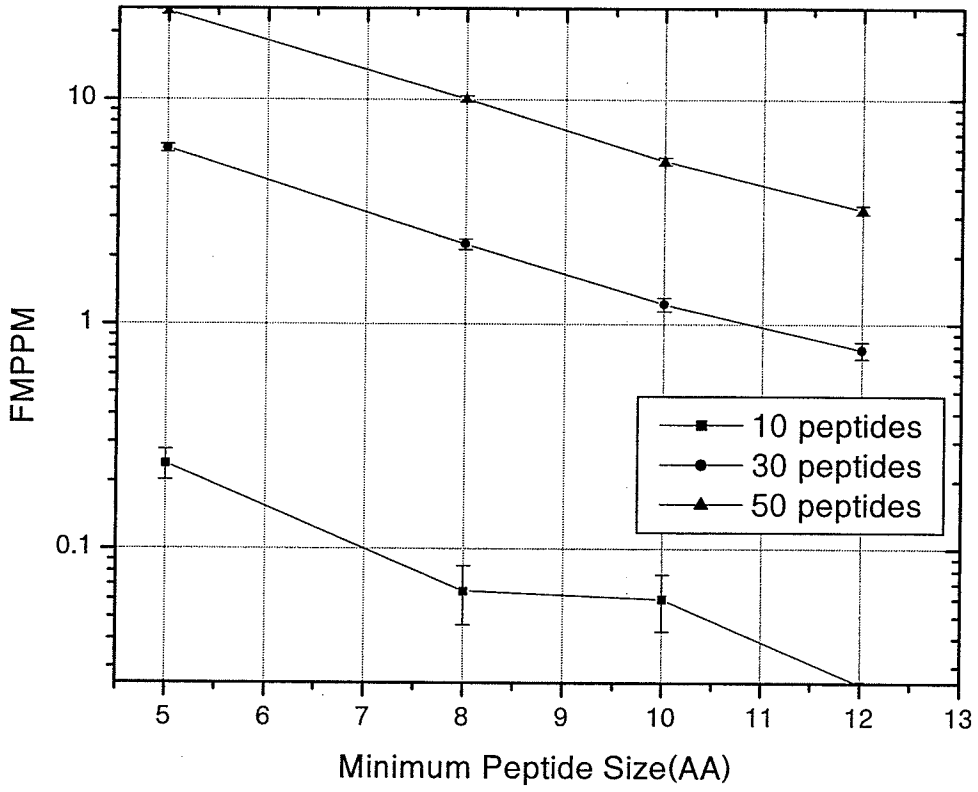

Figure 4.14: FMPPM vs. minimum peptide size

4.7 Molecular Weight Constraints

The molecular weight of the intact protein can be entered as a parameter into a protein search engine. Candidate matches outside a specified range of molecular weights are excluded from the results, and in this way many false matches can be eliminated. A short simulation was performed to investigate this.

One peptide map of with molecular weight of approximately 20kDa was used in a search against *E. coli*. Potential matches were screened based on whether or not they fell inside a range of 20kDa ± x; x denotes the values on the x-axis of fig (4.15).

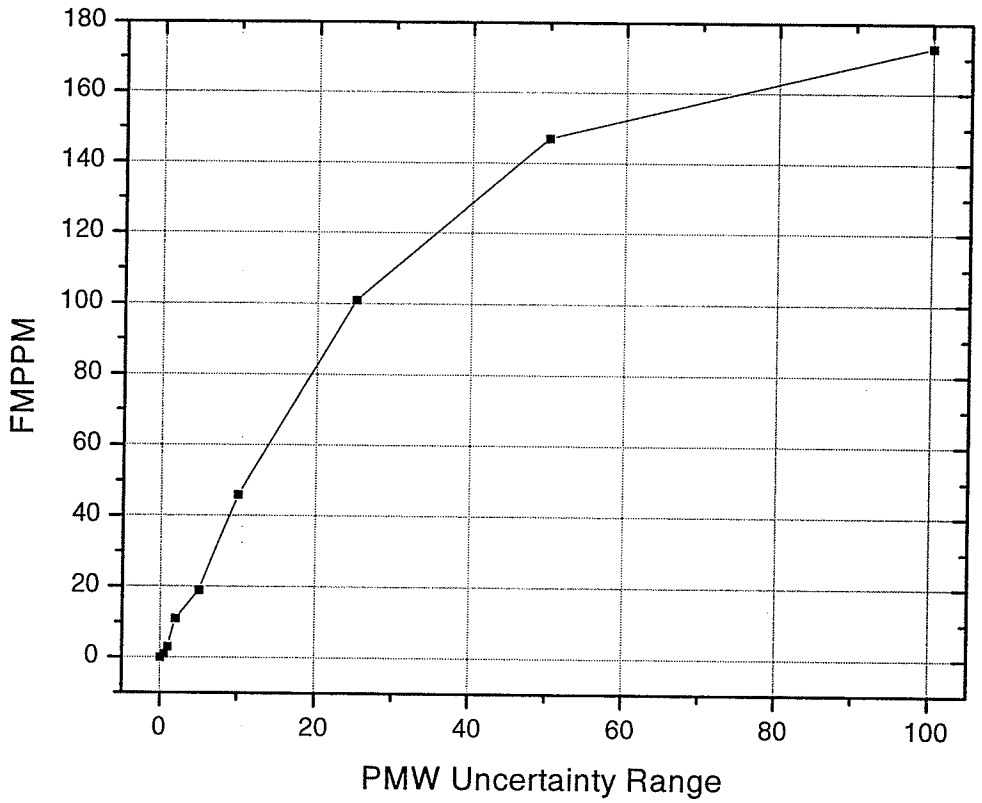

Figure 4.15: FMPPM vs PMW uncertainty range

This approximates a logarithmic curve. High confidence in the intact protein molecular weight (±5kDa) results in almost a 10-fold reduction in false matches over an unrestricted search.

Chapter 5

Conclusions and Future Work

5.1 Summary of Objectives

Generation of random peptides to be used as source data for PDB searches

This facet of the project was accomplished by using a pseudo random number generator to select random peptides from a pool of approximately 450,000 proteins covering most existing proteomes.

Exploration of parameter space

Parameter space was explored in increments through the following parameter ranges:

- MA: 0-100, 400 (in a study of *E. coli*)
- nRMP: 0-6 and 10
- Random Peptide Map Size: 5 sizes: 5,10,30,50,80 through 2-4 nRMP
- Missed Cleavages: 0-4
- Minimum Peptide Size: 5,8,10,12
- Molecular Weight Ranges: 20±(10-1000)Da

Development of a cluster computer system

A workstation cluster was developed and implemented in-house on non-dedicated computer. A central control program directed the workstation cluster through a TCP/IP interface.

Development of a generalized search engine that is adaptable for future applications.

A search algorithm was developed that is easily adaptable to future applications.

5.2 Discussion

It is important to understand the interrelationship of search parameters in a protein database search. By analyzing each parameter separately while holding all others constant, information has been gathered that is useful in designing improved algorithms for handling searches, and is useful in programming other tools for the bioinformaticians.

Mass accuracy is one such parameter that is often considered as an option for increasing confidence in a match. The plots that result are always sigmoid in shape, with the largest increase in false matches occurring in the 50ppm range. In the range of 0-10 ppm, there is much less difference in false matches, due to the fact that the curve is much more horizontal. For 2nRMP, 8MINAA, 0MC parameters, the false matches dip below 1 FMPPM at approximately 7ppm. At 50ppm, false matches increase to 30 FMPPM. This is an unacceptable value for a confident identification of an experimental peptide map. A good confidence level would be <10 FMPPM. Increasing the nRMP parameter to 3 in this case would yield ~3 false matches.

An interesting phenomenon is a plateau from 100ppm to approximately 500ppm, which is due to the fact that the mass distribution of peptides is non-uniform, with 1 Da gaps between groups of peptides.

The most important parameter in terms of the variance in FMPPM is the number of required matching peptides. There exists a significant difference in varying the parameter between 1 and 2, while keeping all others constant. There is a 10-fold drop in FMPPM for each extra nRMP beyond one. In simulations an effective elimination of false matches for 2 nRMP was at a MA value of 5 ppm, while with 6nRMP, the MA value becomes 80ppm.

A study was performed on the influence on the peptide map size. Five maps were generated between 5 and 80 peptides. More peptides leads to an increase in FMPPM, however, there is a slight decrease if the value of nRMP was increased by one at 50 and 80. In fact, there is no real benefit from running larger peptide maps over smaller ones. The overall point here is that running large peptide maps with standard mass confidence will yield little confidence in a result due to a large amount of coincidence matching. While the chance of including the correct match in the list is more likely with larger maps, false matches will obscure that same match. 10 or fewer peptides with high mass accuracy (such as derived using a technique such as MS/MS) are sufficient to locate the correct match and keep false matches at a minimum.

When FMPPM were plotted against missed cleavages, the results were approximately linear. The overall effect of increasing missed cleavages is not drastic, with a doubling or less per unitary increase in missed cleavages.

Some highlight conditions for producing <1 false matches include 3 or more nRMP at 30ppm or <10 ppm accuracy at 2 nRMP). However, many combinations are possible using the available data.

One would expect that when performing searches using larger experimental peptides, the number of false matches would be reduced. In fact, examining figure (4.8), one notices an exponential decrease in peptides for larger mass ranges. Thus, by increasing the peptide size leads to a corresponding exponential decrease in matches.

Finally, investigations were made of the effects of a confidence value for the intact protein molecular weight. False matches are not eliminated even with a very confident value of ΔM, however, with a ± 5 kDa accuracy, false matches are reduced by approximately a factor of 10. It may not always be possible to isolate the weight of the parent protein to this degree in experimental work, when chemical modifications such as glycosylation hamper accurate parent weight determination.

This project is an extension of recent work published by bioinformaticians[14] in the last few years. While it was necessary to program a search engine from the ground up, it provided an opportunity to fully understand the underlying principles, and to approach the problem from a completely different programming stance.

The investigations here should prove useful on protein database search algorithms. Using the program framework currently in place, and the ability to perform quick simulations using almost any combinations of search parameters, the work done here can be extended in any direction to investigate specific traits or characteristics of PDS parameter space.

[14] [16], [17], [23], and most notably [31]

Bibliography

---

[1] http://pir.georgetown.edu/

[2] http://bmbsgi11.leeds.ac.uk/bmb5dp/owl.html

[3] http://www.embl-heidelberg.de/

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

[5] http://www.ddbj.nig.ac.jp/

[6] http://www.hgmp.mrc.ac.uk/Bioinformatics/Webapp/mowse/

[7] http://www.matrixscience.com

[8] http://prospector.ucsf.edu/

[9] http://www.expasy.ch

[10] http://www.proteomtrics.com

[11] http://service.proteometrics.com/prowl/profound.html

[12] J. D. Watson and F. H. C. Crick, A Structure for Deoxyribose Nucleic Acid *Nature* 171, 737-738 (1953)

[13] Dayhoff, M. Schwartz, R. M. and Orcutt, B. C. (1978). Atlas of Protein Structure 1978

[14] Bleasby, Alan J., Wootton, John C., Construction of Validated, Non-Redundant Composite Protein Sequence Database, *Protein Engineering*, vol. 3, no. 3, 153-159, 1990

[15] Bell, G. (1992): Ultra-Computer: A Teraflop before its Time. *Comm. ACM*, 35(8), 26-47

[16] Pappin, D.J.C. *et al.*(1993): Rapid Identification of Proteins by Peptide-Mass Fingerprinting *Current Biology*, vol. 3 No 6, 327-332

[17] Mann, M. et al(1993): Use of Mass Spectrometric Molecular Weight Information to Identify Proteins in Sequence Databases, *Biol. Mass Spectrometry*, Vol 22, 338-345 (1993)

[18] Karas, Hillenkamp, *Anal. Chem*, 60, 2299 (1988)

[19] Hillenkamp, *Anal Chem* 63, 1193A (1991)

[20] Fenn, Electrospray Ionization for Large Biomolecules, *Science* 246, 64-71

[21] Jensen, Ole N., Podtelejnikov, Alexandre, Mann, Matthias: Delayed Extraction Improves Specificity in Database Searches by Matrix-Assisted Laser Desorption/Ionization Peptide Maps, *Rapid Comm. in Mass Spectrometry*, vol. 10, 1371-1378 (1996)

[22] Chernushevich, Igor; Ens, Werner; Standing, K.G.; Orthogonal Injection TOF-MS for Analysing Biomolecules, *Anal Chem News and Features,* 452-461, 1991

[23] Zhang, Wenzhu; Chait, Brian; Profound: An Expert System for Protein Identification Using Mass Spectrometric Peptide Mapping Information, *Anal Chem.* 2000, 72, 2482-2489

[24] Barker, Winona; Garavelli, John S.; et al. Protein Information Resource: a community resource for expert annotation of protein data, *Nucleic Acids Research*, vol. 29, no. 1, 2001

[25] Fenyö, David, Identifying the proteome: software tools, *Current Opinion in Biotechnology*, 11, 391-395, 2000

[26] Sidhu, Khushwant; Sangvanich, Polkit, Bioinformatic Assessment of Mass Spectrometric Chemical Derivatisation Techniques for Proteome Database Searching, in press (submitted to Proteomics in 2001)

[27] Cao, Ping; Moini, Mehdi, Capillary Electrophoresis/Electrospray Ionization High Mass Accuracy Time-of-Flight Mass Spectrometry for Protein Identification Using Peptide Mapping, *Rapid Comm. in Mass Spectrometry*, 12, 864-870 (1998)

[28] Yates III, John R. ; Eng, J.K. ; McCormack, A.L. ; Schieltz, David, Method to Correlate Tandem Mass Spectra of Modified Peptides to Amino Acid Sequences in the Protein Database, *Anal. Chem.*, 67, 1426-1436, 1995

[29] D. R. Goodlett, J.E. Bruce, G.A. Anderson, B. Rist, L. Pasa-Tolic, O. Fiehn, R. Smith, and R. Aebersold, Protein Identification with a Single Accurate Mass of a Cysteine-Containing Peptide and Constrained Database Searchingm *Anal. Chem* , 71, 1112-1118, 2000

[30] S. Sechi and B. T. Chait, Modification of Cysteine Residues by Alkylation: A Tool in Peptide Mapping and Protein Identification, *Anal. Chem.*, 70, 5150-5158, 1998

[31] J. Eriksson, B. T. Chait, and D. Fenyo, A Statistical Basis for Testing the Significance of Mass Spectrometric Protein Identification Results, *Anal Chem.*, 72, 999-1005, 2000

Appendix A

Database File Formats

GENBANK File Format

```
LOCUS       SCU49845    5028 bp    DNA             PLN       21-JUN-1999
DEFINITION  Saccharomyces cerevisiae TCP1-beta gene, partial cds, and Axl2p
            (AXL2) and Rev7p (REV7) genes, complete cds.
ACCESSION   U49845
VERSION     U49845.1  GI:1293613
KEYWORDS    .
SOURCE      baker's yeast.
  ORGANISM  Saccharomyces cerevisiae
            Eukaryota; Fungi; Ascomycota; Hemiascomycetes; Saccharomycetales;
            Saccharomycetaceae; Saccharomyces.
REFERENCE   1  (bases 1 to 5028)
  AUTHORS   Torpey,L.E., Gibbs,P.E., Nelson,J. and Lawrence,C.W.
  TITLE     Cloning and sequence of REV7, a gene whose function is required for
            DNA damage-induced mutagenesis in Saccharomyces cerevisiae
  JOURNAL   Yeast 10 (11), 1503-1509 (1994)
  MEDLINE   95176709
REFERENCE   2  (bases 1 to 5028)
  AUTHORS   Roemer,T., Madden,K., Chang,J. and Snyder,M.
  TITLE     Selection of axial growth sites in yeast requires Axl2p, a novel
            plasma membrane glycoprotein
  JOURNAL   Genes Dev. 10 (7), 777-793 (1996)
  MEDLINE   96194260
REFERENCE   3  (bases 1 to 5028)
  AUTHORS   Roemer,T.
  TITLE     Direct Submission
  JOURNAL   Submitted (22-FEB-1996) Terry Roemer, Biology, Yale University, New
            Haven, CT, USA
FEATURES             Location/Qualifiers
     source          1..5028
                     /organism="Saccharomyces cerevisiae"
                     /db_xref="taxon:4932"
                     /chromosome="IX"
                     /map="9"
     CDS             <1..206
```

```
                /codon_start=3
                /product="TCP1-beta"
                /protein_id="AAA98665.1"
                /db_xref="GI:1293614"
                /translation="SSIYNGISTSGLDLNNGTIADMRQLGIVESYKLKRAVVSSASEA
                AEVLLRVDNIIRARPRTANRQHM"
     gene       687..3158
                /gene="AXL2"
     CDS        687..3158
                /gene="AXL2"
                /note="plasma membrane glycoprotein"
                /codon_start=1
                /function="required for axial budding pattern of S.
                cerevisiae"
                /product="Axl2p"
                /protein_id="AAA98666.1"
                /db_xref="GI:1293615"
                /translation="MTQLQISLLLTATISLLHLVVATPYEAYPIGKQYPPVARVNESF
                TFQISNDTYKSSVDKTAQITYNCFDLPSWLSFDSSSRTFSGEPSSDLLSDANTTLYFN
                VILEGTDSADSTSLNNTYQFVVTNRPSISLSSDFNLLALLKNYGYTNGKNALKLDPNE
                VFNVTFDRSMFTNEESIVSYYGRSQLYNAPLPNWLFFDSGELKFTGTAPVINSAIAPE
                TSYSFVIIATDIEGFSAVEVEFELVIGAHQLTTSIQNSLIINVTDTGNVSYDLPLNYV
                YLDDDPISSDKLGSINLLDAPDWVALDNATISGSVPDELLGKNSNPANFSVSIYDTYG
                DVIYFNFEVVSTTDLFAISSLPNINATRGEWFSYYFLPSQFTDYVNTNVSLEFTNSSQ
                DHDWVKFQSSNLTLAGEVPKNFDKLSLGLKANQGSQSQELYFNIIGMDSKITHSNHSA
                NATSTRSSHHSTSTSSYTSSTYTAKISSTSAAATSSAPAALPAANKTSSHNKKAVAIA
                CGVAIPLGVILVALICFLIFWRRRRENPDDENLPHAISGPDLNNPANKPNQENATPLN
                NPFDDDASSYDDTSIARRLAALNTLKLDNHSATESDISSVDEKRDSLSGMNTYNDQFQ
                SQSKEELLAKPPVQPPESPFFDPQNRSSSVYMDSEPAVNKSWRYTGNLSPVSDIVRDS
                YGSQKTVDTEKLFDLEAPEKEKRTSRDVTMSSLDPWNSNISPSPVRKSVTPSPYNVTK
                HRNRHLQNIQDSQSGKNGITPTTMSTSSSDDFVPVKDGENFCWVHSMEPDRRPSKKRL
                VDFSNKSNVNVGQVKDIHGRIPEML"
     gene       complement(3300..4037)
                /gene="REV7"
     CDS        complement(3300..4037)
                /gene="REV7"
                /codon_start=1
                /product="Rev7p"
                /protein_id="AAA98667.1"
                /db_xref="GI:1293616"
                /translation="MNRWVEKWLRVYLKCYINLILFYRNVYPPQSFDYTTYQSFNLPQ
                FVPINRHPALIDYIEELILDVLSKLTHVYRFSICIINKKNDLCIEKYVLDFSELQHVD
                KDDQIITETEVFDEFRSSLNSLIMHLEKLPKVNDDTITFEAVINAIELELGHKLDRNR
                RVDSLEEKAEIERDSNWVKCQEDENLPDNNGFQPPKIKLTSLVGSDVGPLIIHQFSEK
                LISGDDKILNGVYSQYEEGESIFGSLF"
BASE COUNT    1510 a   1074 c    835 g    1609 t
ORIGIN
      1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
     61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
    121 ctgcatctga agccgctgaa gttctactaa gggtggataa catcatccgt gcaagaccaa
    181 gaaccgccaa tagacaacat atgtaacata tttaggatat acctcgaaaa taataaaccg
    241 ccacactgtc attattataa ttagaaacag aacgcaaaaa ttatccacta tataattcaa
    301 agacgcgaaa aaaaaagaac aacgcgtcat agaacttttg gcaattcgcg tcacaaataa
```

```
361 attttggcaa cttatgtttc ctcttcgagc agtactcgag ccctgtctca agaatgtaat
421 aatacccatc gtaggtatgg ttaaagatag catctccaca acctcaaagc tccttgccga
...
```

SWISS-PROT File Format

```
ID  ADP1_YEAST     STANDARD;      PRT;  1049 AA.
AC  P25371;
DT  01-MAY-1992 (Rel. 22, Created)
DT  01-MAY-1992 (Rel. 22, Last sequence update)
DT  16-OCT-2001 (Rel. 40, Last annotation update)
DE  Probable ATP-dependent permease precursor.
GN  ADP1 OR YCR011C OR YCR11C OR YCR105.
OS  Saccharomyces cerevisiae (Baker's yeast).
OC  Eukaryota; Fungi; Ascomycota; Saccharomycotina; Saccharomycetes;
OC  Saccharomycetales; Saccharomycetaceae; Saccharomyces.
OX  NCBI_TaxID=4932;
RN  [1]
RP  SEQUENCE FROM N.A.
RX  MEDLINE=92160395; PubMed=1789009; [NCBI, ExPASy, EBI, Israel, Japan]
RA  Purnelle B., Skala J., Goffeau A.;
RT  "The product of the YCR105 gene located on the chromosome III from
RT  Saccharomyces cerevisiae presents homologies to ATP-dependent
RT  permeases.";
RL  Yeast 7:867-872(1991).
RN  [2]
RP  SEQUENCE FROM N.A.
RX  MEDLINE=92327849; PubMed=1626432; [NCBI, ExPASy, EBI, Israel, Japan]
RA  Skala J., Purnelle B., Goffeau A.;
RT  "The complete sequence of a 10.8 kb segment distal of SUF2 on the
RT  right arm of chromosome III from Saccharomyces cerevisiae reveals
RT  seven open reading frames including the RVS161, ADP1 and PGK genes.";
RL  Yeast 8:409-417(1992).
CC  -!- SUBCELLULAR LOCATION: Integral membrane protein (Potential).
CC  -!- SIMILARITY: BELONGS TO THE ABC TRANSPORTER FAMILY. MDR SUBFAMILY.
CC  ---------------------------------------------------------------------
CC  This SWISS-PROT entry is copyright. It is produced through a collaboration
CC  between the Swiss Institute of Bioinformatics and the EMBL outstation -
CC  the European Bioinformatics Institute. There are no restrictions on its
CC  use by non-profit institutions as long as its content is in no way
CC  modified and this statement is not removed. Usage by and for commercial
CC  entities requires a license agreement (See http://www.isb-sib.ch/announce/
CC  or send an email to license@isb-sib.ch).
CC  ---------------------------------------------------------------------
DR  EMBL; X59720; CAA42328.1; -. [EMBL / GenBank / DDBJ] [CoDingSequence]
DR  PIR; S19421; S19421.
DR  PIR; S40914; S40914.
DR  SGD; S0000604; ADP1.
DR  YPD; ADP1.
```

DR InterPro; IPR003593; AAA.
DR InterPro; IPR003439; ABC_transportr.
DR InterPro; IPR001687; ATP_GTP_A.
DR InterPro; Graphical view of domain structure.
DR Pfam; PF00005; ABC_tran; 1.
DR ProDom; PD000006; ABC_transportr; 1.
DR ProDom [Domain structure / List of seq. sharing at least 1 domain]
DR SMART; SM00382; AAA; 1.
DR SMART; SM00001; EGF_like; 1.
DR PROSITE; PS00211; ABC_TRANSPORTER; 1.
DR BLOCKS; P25371.
DR ProtoMap; P25371.
DR PRESAGE; P25371.
DR DIP; P25371.
DR ModBase; P25371.
DR GeneCensus; YCR011C.
DR SWISS-2DPAGE; GET REGION ON 2D PAGE.
KW ATP-binding; Transmembrane; Glycoprotein; Transport; Signal.
FT SIGNAL 1 25 POTENTIAL.
FT CHAIN 26 1049 PROBABLE ATP-DEPENDENT PERMEASE.
FT NP_BIND 423 430 ATP (BY SIMILARITY).
FT TRANSMEM 325 345 POTENTIAL.
FT TRANSMEM 464 481 POTENTIAL.
FT TRANSMEM 794 814 POTENTIAL.
FT TRANSMEM 829 849 POTENTIAL.
FT TRANSMEM 878 898 POTENTIAL.
FT TRANSMEM 910 930 POTENTIAL.
FT TRANSMEM 938 958 POTENTIAL.
FT TRANSMEM 1001 1021 POTENTIAL.
FT TRANSMEM 1025 1045 POTENTIAL.
FT CARBOHYD 50 50 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 114 114 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 165 165 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 221 221 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 815 815 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 935 935 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 960 960 N-LINKED (GLCNAC...) (POTENTIAL).
FT CARBOHYD 971 971 N-LINKED (GLCNAC...) (POTENTIAL).
SQ SEQUENCE 1049 AA; 117231 MW; ABC9CE54BCFDF6A3 CRC64;
MGSHRRYLYY SILSFLLLSC SVVLAKQDET PFFEGTSSKN SRLTAQDKGN DTCPPCFNCM
LPIFECKQFS ECNSYTGRCE CIEGFAGDDC SLPLCGGLSP DESGNKDRPI RAQNDTCHCD
NGWGGINCDV CQEDFVCDAF MPDPSIKGTC YKNGMIVDKV FSGCNVTNEK ILQILNGKIP
QITFACDKPN QECNFQFWID QLESFYCGLS DCAFEYDLEQ NTSHYKCNDV QCKCVPDTVL
CGAKGSIDIS DFLTETIKGP GDFSCDLETR QCKFSEPSMN DLILTVFGDP YITLKCESGE
CVHYSEIPGY KSPSKDPTVS WQGKLVLALT AVMVLALFTF ATFYISKSPL FRNGLGSSKS
PIRLPDEDAV NNFLQNEDDT LATLSFENIT YSVPSINSDG VEETVLNEIS GIVKPGQILA
IMGGSGAGKT TLLDILAMKR KTGHVSGSIK VNGISMDRKS FSKIIGFVDQ DDFLLPTLTV
FETVLNSALL RLPKALSFEA KKARVYKVLE ELRIIDIKDR IIGNEFDRGI SGGEKRRVSI
...

PIR File Format

ENTRY DEBYA2 #type complete **ProClass View of DEBYA2**
TITLE alcohol dehydrogenase (EC 1.1.1.1) 2 - yeast (Saccharomyces cerevisiae)
ALTERNATE_NAMES protein YM9952.05c; protein YMR303c
ORGANISM #formal_name Saccharomyces cerevisiae
#cross-references taxon:4932
DATE 25-Feb-1985 #sequence_revision 25-Feb-1985 #text_change 21-Jul-2000
ACCESSIONS A00340; S38796; S53973
REFERENCE A00340
#authors Russell, D.W.; Smith, M.; Williamson, V.M.; Young, E.T.
#journal J. Biol. Chem. (1983) 258:2674-2682
#title Nucleotide sequence of the yeast alcohol dehydrogenase II gene.
#cross-references MUID:83109119
#accession A00340
##molecule_type DNA
##residues 1-348 ##label RUS
##cross-references EMBL:V01293; NID:g3344; PIDN:CAA24602.1; PID:g600021
REFERENCE S38795
#authors Young, T.; Williamson, V.; Taguchi, A.; Smith, M.; Sledziewski, A.; Russell, D.; Osterman, J.; Denis, C.; Cox, D.; Beier, D.
#journal Basic Life Sci. (1982) 19:335-361
#title The alcohol dehydrogenase genes of the yeast, Saccharomyces cerevisiae: isolation, structure, and regulation.
#cross-references MUID:82160017
#accession S38796
##molecule_type DNA
##residues 1-15,'H',17-30,'A',32-348 ##label YOU
##cross-references EMBL:M38457; NID:g171028; PIDN:AAA34411.1; PID:g171029
REFERENCE S53969
#authors Connor, R.; Churcher, C.M.
#submission submitted to the EMBL Data Library, April 1995
#accession S53973
##molecule_type DNA
##residues 1-348 ##label CON
##cross-references EMBL:Z49212; NID:g798940; PIDN:CAA89136.1; PID:g798945; GSPDB:GN00013; MIPS:YMR303c
GENETICS
#gene SGD:ADH2; ADR2; MIPS:YMR303c
##cross-references SGD:S0004918; MIPS:YMR303c
#map_position 13R
COMPLEX homotetramer
FUNCTION

#description catalyzes the oxidation of primary and secondary alcohols to aldehydes and ketones, respectively, by NAD+
#pathway alcohol degradation
CLASSIFICATION SF000091
#superfamily alcohol dehydrogenase; long-chain alcohol dehydrogenase homology
KEYWORDS alcohol metabolism; homotetramer; metalloprotein; NAD; oxidoreductase; zinc
FEATURE
29-336 #domain long-chain alcohol dehydrogenase homology #label LADH\
173-202 #region beta-alpha-beta NAD nucleotide-binding fold\
44,67,154 #binding_site zinc, catalytic (Cys, His, Cys) #status predicted\
98,101,104,112 #binding_site zinc, noncatalytic (Cys) #status predicted
SUMMARY #length 348 #molecular_weight 36732

SEQUENCE

```
        5       10       15       20       25       30
 1 M S I P E T Q K A I I F Y E S N G K L E H K D I P V P K P K
31 P N E L L I N V K Y S G V C H T D L H A W H G D W P L P T K
61 L P L V G G H E G A G V V V G M G E N V K G W K I G D Y A G
91 I K W L N G S C M A C E Y C E L G N E S N C P H A D L S G Y
121 T H D G S F Q E Y A T A D A V Q A A H I P Q G T D L A E V A
151 P I L C A G I T V Y K A L K S A N L R A G H W A A I S G A A
181 G G L G S L A V Q Y A K A M G Y R V L G I D G G P G K E E L
211 F T S L G G E V F I D F T K E K D I V S A V V K A T N G G A
241 H G I I N V S V S E A A I E A S T R Y C R A N G T V V L V G
271 L P A G A K C S S D V F N H V V K S I S I V G S Y V G N R A
301 D T R E A L D F F A R G L V K S P I K V V G L S S L P E I Y
331 E K M E K G Q I A G R Y V V D T S K
```

Appendix B

Amino Acid Residue Masses

Amino Acid	3-letter code	1-letter code	Monoisotopic Mass(Da)	Average Mass(Da)
Glycine	Gly	G	57.0215	57.052
Alanine	Ala	A	71.0371	71.079
Serine	Ser	S	87.0320	87.078
Proline	Pro	P	97.0528	97.117
Valine	Val	V	99.0684	99.133
Threonine	Thr	T	101.0477	101.105
Cysteine	Cys	C	103.0092	130.144
Isoleucine	Ile	I	113.0841	113.160
Leucine	Leu	L	113.0841	113.160
Asparagine	Asn	N	114.0429	114.104
Aspartic Acid	Asp	D	115.0270	115.089
Glutamine	Gln	Q	128.0586	128.131
Lysine	Lys	K	128.0950	128.174
Glutamic Acid	Glu	E	129.0426	129.116
Methionine	Met	M	131.0405	131.198
Histidine	His	H	137.0589	137.142
Phenylalanine	Phe	F	147.0684	147.177
Arginine	Arg	R	156.1011	156.188
Tyrosine	Tyr	Y	163.0633	163.17
Tryptophan	Trp	W	186.0793	186.213

Appendix C

Pepstat and Control GUI's

Pepstat

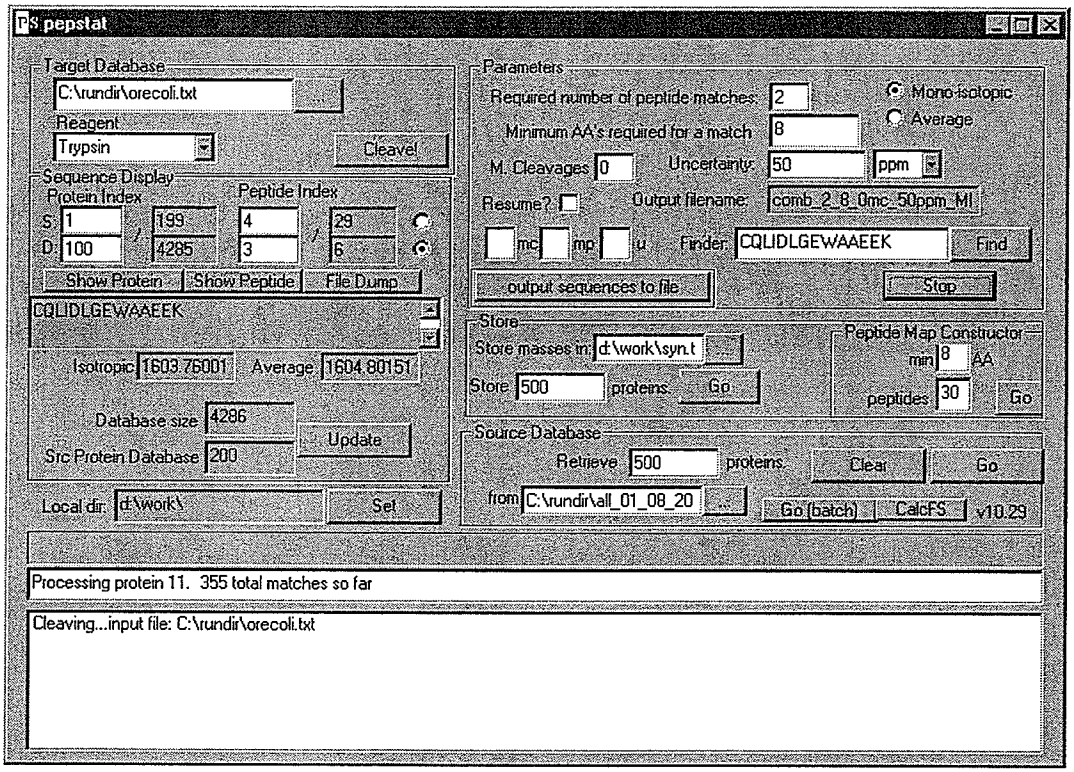

Figure C.1: Pepstat GUI

Sections

Top Left: *Target Database*
Bottom Right: *Source Database*
Middle Right: *Random Peptide Map Constructor*
Top Right: *Parameters*
Middle Right: *Reserved for examining sequence information*
Bottom: *Dialog Window*

Control

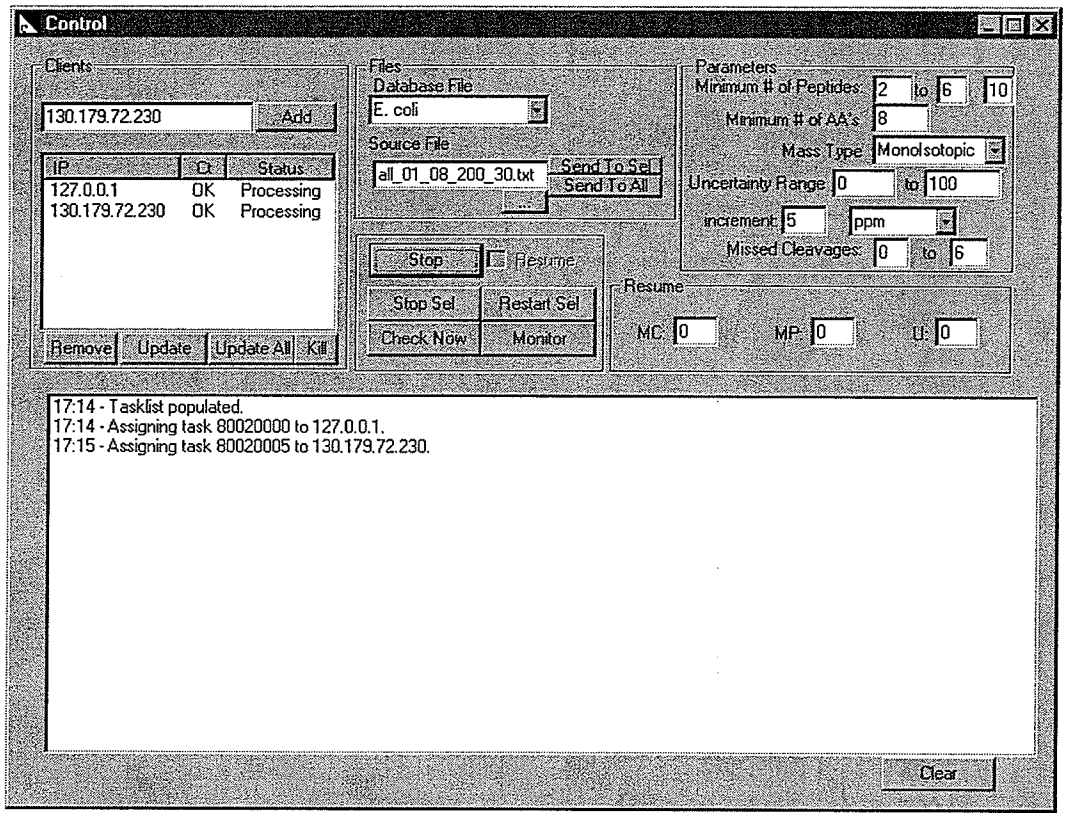

Figure C.2: Control GUI

Sections

Top Left: *Client List*
Top Middle: *Database File Selection and File Transfer Option*
Top Right: *Parameter List (with iteration parameters)*
Middle: *Control Buttons*
Bottom: *Dialog Window*