A Holistic Computational Approach to Boosting the Performance of Protein Search Engines

Majdi Ahmad Mosa Maabreh
Western Michigan University

Follow this and additional works at: https://scholarworks.wmich.edu/dissertations

Part of the Computer Sciences Commons

Recommended Citation
https://scholarworks.wmich.edu/dissertations/3221

This Dissertation-Open Access is brought to you for free and open access by the Graduate College at ScholarWorks at WMU. It has been accepted for inclusion in Dissertations by an authorized administrator of ScholarWorks at WMU. For more information, please contact wmu-scholarworks@wmich.edu.
A HOLISTIC COMPUTATIONAL APPROACH TO BOOSTING THE PERFORMANCE OF PROTEIN SEARCH ENGINES

by

Majdi Ahmad Mosa Maabreh

A dissertation submitted to the Graduate College in partial fulfillment of the requirements for the degree of Doctor of Philosophy
Computer Science
Western Michigan University
April 2018

Doctoral Committee:

Ajay Gupta, Ph.D., Chair
Ala Al-Fuqaha, Ph.D.
James Springstead, Ph.D.
Alexey Nesvizhskii, Ph.D.
Izzat Alsmadi, Ph.D.
Despite availability of several proteins search engines, due to the increasing amounts of MS/MS data and database sizes, more efficient data analysis and reduction methods are important. Improving accuracy and performance of protein identification is a main goal in the community of proteomic research. In this research, a holistic solution for improvement in search performance is developed.

Most current search engines apply the SEQUEST style of searching protein databases to define MS/MS spectra. SEQUEST involves three main phases: (i) Indexing the protein databases, (ii) Matching and Ranking the MS/MS spectra and (iii) Filtering the matches and reporting the final proteins. Technical analysis of each phase resulted in several potential improvements that have been implemented in a holistic, multidimensional approach. This dissertation focuses on challenges and limitations of the current protein search engines, while providing solutions to address these problems. Primarily, indexing and the searching phases are optimized.

This dissertation describes the indexing phase in a commonly used search engine and provides an alternative solution to code and data structure optimizations, making indexing more efficient and less computationally intensive. This method may be applied in metaproteomics.
studies, where large protein databases are typically used in identifying proteins from complex samples and individual organisms are not identified in advance. In the searching phase, a deep-learning algorithm and different shallow learning algorithms are tested to reduce computation load of the matching process. The main objective is to reduce unnecessary load introduced by “possibly irrelevant” MS/MS spectra. The deep learning algorithm may be especially useful when the protein(s) of interest are in lower cellular or tissue concentration, while the other algorithms may be more useful for concentrated or more highly expressed proteins. To improve the accuracy of identification and to adjust searching parameters, particle swarm optimization (PSO) is utilized to configure the search engine parameters, resulting in optimization of the matching process. Experimentally, the PSO model shows encouraging results and covers some limitations of the previous works in parameters configuration.

Due to diversity of search engine coverage and overlapping results, it is proposed to combine results of multiple search engines to increase reliability of identification. However, despite straightforward implementations on cloud or distributed environments, transfer of MS/MS spectra among the systems’ various units is a major concern and should be carefully handled. The impact of transferring the raw spectra to the computing nodes is presented in this research, and two different approaches using peaks sampling and machine learning have been developed as spectra reduction methods. The results of both solutions show significant MS/MS size reduction, thereby mitigating unnecessary communication and computation, while also potentially reducing cost in cloud-based pay-as-you-go environments.
ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious and the Most Merciful. All praises are due to Allah for his blessings in completing this dissertation.

This dissertation is the peak of my exciting Ph.D. journey which started in 2014. During this time, I have received support from a great number of individuals. This journey would not have been possible without the support of my family, professors and friends.

I would like to express my special appreciation to my parents for showing faith in me and the sacrifices they did to shape my life.

Special thanks and deepest gratitude to my wife, Arwa, who gave support and help during my graduate studies. She has substituted my role in many times and taken the responsibility of our family. Along with her, I would like to acknowledge my beloved daughter Lotas and my beloved sons Ameer and Adam for being good kids. They know their dad is a student and he needs more time now eventually to make their life better. Thanks my family for your smiles and prayers. You always find a way to cheer me up.

I would also like to thank my brother Mohammad for his continuing support and encouragement. I will not forget my dearest sisters Rana, Reem and Enas for their prayers and nice words of support.

My parents-in-law, sisters-in-law and brothers-in-law have been all wonderful, indeed, and I really appreciate their support.

I would also like to express my special appreciation and thanks to my advisor Prof. Ajay Gupta for encouraging and supporting me to be a researcher. I would also like to thanks the
committee members for their constructive feedback and brilliant comments and suggestions that make this research work more impactful.

I will not forget to thank my sponsor, The Hashemite University, My research was greatly facilitated by their support.

Finally, my thanks go to all the people who have supported me to complete the research work directly or indirectly.

Majdi Ahmad Mosa Maabreh
TABLE OF CONTENTS

ACKNOWLEDGEMENTS ........................................................................................................ ii
LIST OF TABLES .................................................................................................................. x
LIST OF FIGURES ............................................................................................................. xii

CHAPTER
1 INTRODUCTION ............................................................................................................. 1

1.1 Mass Spectrometry-based Proteomics ................................................................. 1
1.2 Searching Against Proteins’ Database ..................................................................... 2
1.3 Proteins Database Search Engines ......................................................................... 4
1.4 The Choice of Databases in Proteomics ............................................................... 6

1.4.1 The Protein Databases for Single Organism Experiments: the Load of Extra Sequences .......................................................................................... 7
1.4.2 Large Protein Databases in Metaproteomics ...................................................... 8

1.5 Computational Challenges from the Size of Observed Spectra in Proteomics ................................................................................................................. 10

1.5.1 Computation Speed and Throughput ................................................................. 10
1.5.2 Memory Footprint ............................................................................................ 12
1.5.3 File Transfer ................................................................................................... 12
1.5.4 The Flood of Spectra in HPC Solutions ........................................................... 13
1.5.5 Storage and Backup ....................................................................................... 13
1.5.6 Applications of Machine Learning ................................................................... 14

1.6 Major Issues in the Current MS/MS Search Engines ............................................. 14
TABLE OF CONTENTS—Continued

CHAPTER

1.6.1 Target-Decoy Databases Indexing .......................................................... 15

1.6.2 Searching Massive MS/MS against another Massive Databases .......................................................... 16

1.6.3 Searching Optimization ........................................................................... 17

1.6.4 The Reliability of the PSMs and the Overlap among Multiple Search Engines ........................................................................ 18

1.7 Dissertation’s Contributions .............................................................................. 20

1.7.1 Chapter 2: A Multithreading and Hashing Technique for Indexing Target-Decoy Peptides Databases. ................................................... 20

1.7.2 Chapter 3: Deep vs. Shallow Learning-based Filters of MS/MS Spectra in Support of Protein Search Engines ........................................ 21

1.7.3 Chapter 4: Optimizing Protein Search Engines using Particle Swarm Optimization ................................................................. 22

1.7.4 Chapter 5: Deep Learning–based MSMS Spectra Reduction in Support of Running Multiple Protein Search Engines on Cloud ............................................................. 23

1.7.5 Chapter 6: Towards Centralized MS/MS Spectra Preprocessing in Cloud-based Ensemble Solutions of Multiple Search Engines ........................................ 23

References ......................................................................................................... 25

2 A MULTITHREADING AND HASHING TECHNIQUE FOR INDEXING TARGET-DECOY PEPTIDES DATABASES .......................................................... 31

2.1 Summary ..................................................................................................... 31

2.2 Introduction .................................................................................................. 32

2.2.1 Target-Decoy Database Strategy ................................................................. 34

2.2.2 Target-Decoy Indexing in Crux-Tide ................................................................. 36

2.3 Sequential Crux-Tide Indexer ........................................................................ 37
Table of Contents—Continued

CHAPTER

2.4 Multithreading Hash-based Solution ................................................................. 42
2.5 Code and Memory Optimization ....................................................................... 46
2.6 Experiments Environments and Data .............................................................. 48
2.7 Results and Discussion ...................................................................................... 49
  2.7.1 Code and Memory Optimization Evaluation ........................................... 50
  2.7.2 Parallel Solution Evaluation .................................................................... 52
2.8 Conclusion ......................................................................................................... 56

References ..................................................................................................................... 57

3 DEEP VS. SHALLOW LEARNING-BASED FILTERS OF MS/MS SPECTRA IN SUPPORT OF PROTEIN SEARCH ENGINES ................................................................. 59

3.1 Summary ............................................................................................................ 59
3.2 Introduction ........................................................................................................ 60
3.3 Background ........................................................................................................ 62
3.4 Methods and Materials ....................................................................................... 64
  3.4.1 Testing and Evaluation Datasets ............................................................. 64
  3.4.2 Supervised Machine Learning Algorithms .............................................. 65
  3.4.3 Protein Search Engines ............................................................................ 65
  3.4.4 Datasets for Machine Learning ............................................................... 66
  3.4.5 Models Evaluation ................................................................................... 67
  3.4.6 HEK293 Dataset Processing ................................................................... 68
3.5 Results and Discussion ...................................................................................... 70
  3.5.1 Human01 Dataset .................................................................................... 70
<table>
<thead>
<tr>
<th>CHAPTER</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.2 Human06 Dataset ................................................................. 72</td>
</tr>
<tr>
<td>3.5.3 Human05 Dataset ................................................................. 73</td>
</tr>
<tr>
<td>3.5.4 Mouse Dataset ............................................................... 74</td>
</tr>
<tr>
<td>3.5.5 Soybean Dataset ............................................................. 75</td>
</tr>
<tr>
<td>3.5.6 Rat Dataset ................................................................. 76</td>
</tr>
<tr>
<td>3.5.7 HEK293 Dataset ............................................................. 76</td>
</tr>
<tr>
<td>3.6 Conclusion .............................................................................. 87</td>
</tr>
<tr>
<td>References ..................................................................................... 89</td>
</tr>
</tbody>
</table>

| 4 OPTIMIZING PROTEIN SEARCH ENGINES USING PARTICLE SWARM OPTIMIZATION ......................................................... 91 |
| 4.1 Summary .................................................................................. 91 |
| 4.2 Introduction .......................................................................... 92 |
| 4.3 Particle Swarm Optimization ................................................ 95 |
| 4.4 Methods .................................................................................. 97 |
| 4.4.1 PSO Model ....................................................................... 97 |
| 4.4.2 Data Sets .......................................................................... 98 |
| 4.4.3 Experiments Environment and Comet Parameters ............... 99 |
| 4.5 Results and Discussion .......................................................... 99 |
| 4.6 Related Work ......................................................................... 106 |
| 4.7 Conclusion ............................................................................. 108 |
| References ..................................................................................... 110 |
Table of Contents—Continued

CHAPTER

5 DEEP LEARNING–BASED MSMS SPECTRA REDUCTION IN SUPPORT OF RUNNING MULTIPLE PROTEIN SEARCH ENGINES ON CLOUD ......................................................................................... 112

5.1 Summary .......................................................................................................... 112
5.2 Introduction ...................................................................................................... 113
5.3 Background ...................................................................................................... 115
5.4 Methods and Materials..................................................................................... 118
   5.4.1 Dataset ................................................................................................... 118
   5.4.2 Deep Learning Algorithm ..................................................................... 118
   5.4.3 Protein Search Engines ........................................................................ 119
   5.4.4 Datasets for Machine Learning ............................................................. 120
   5.4.5 Models Evaluation ................................................................................. 121
5.5 Results and Discussion .................................................................................... 122
   5.5.1 The First Experiment ............................................................................. 122
   5.5.2 The Second Experiment ........................................................................ 125
5.6 Conclusion ....................................................................................................... 130

References ................................................................................................................ 132

6 TOWARDS CENTRALIZED MS/MS SPECTRA REPROCESSING: AN EMPIRICAL EVALUATION OF PEPTIDES SEARCH ENGINES USING GROUND TRUTH DATASETS ............................................... 134

6.1 Summary .......................................................................................................... 134
6.2 Introduction ...................................................................................................... 135
6.3 Background ..................................................................................................... 136
# Table of Contents—Continued

## CHAPTER

6.4 Methods ........................................................................................................... 137  
6.4.1 Proteins Search Engines ........................................................................ 137  
6.4.2 Datasets ................................................................................................. 138  
6.4.3 Building Unanimous -Based Dataset .................................................... 139  
6.4.4 Peaks Reduction .................................................................................... 139  

6.5 Results and Discussion .................................................................................... 140  
6.5.1 Unanimous Dataset................................................................................ 140  
6.5.2 Evaluation of Uniform Cut-Off Reduction Technique......................... 141  
6.5.3 Evaluation of Top X% Reduction Technique ....................................... 143  
6.5.4 The Evaluation of Multistage Reduction Techniques ........................... 145  
6.5.5 Evaluating a simple Random Sampling (SRS) as a Reduction Technique ........................................................................................................... 146  
6.5.6 Evaluating Top X% on Different Dataset and Search Engines......................... 148  

6.6 Conclusion ....................................................................................................... 152  
References ................................................................................................................... 154

7 CONCLUSIONS AND FUTURE WORK........................................................................ 156
LIST OF TABLES

1-I: The number of candidates for each spectrum based on three different variables................................................................. 4

1-II: Overlap ratios (Jaccard index) among the search engines using HEK293 Dataset. ................................................................. 19

1-III: Overlap ratios among the search engines using Human Dataset................................. 19

2-I: Datasets used in the experiments. .............................................................................. 49

2-II: Serial execution times and generated peptides. ........................................................ 50

3-I: MS/MS spectra and download information. ................................................................ 64

3-II: The set of features used in this study ......................................................................... 66

3-III: The datasets used in this study. ............................................................................... 67

4-I: MS/MS spectra and the protein databases. ................................................................. 97

4-II: The utilized parameters in our experiments.............................................................. 97

4-III: Comparison between Comet results when using the recommended w by our PSO model and the Param-Medic tool .............................................. 101

4-IV: PSO execution times ............................................................................................. 102

4-V: PSMs from Comet based on tolerance windows recommended by PSO, default windows, and Param-Medic windows. The difference reported is with default parameters....................................................... 103

4-VI: PSMs from Comet based on tolerance windows recommended by PSO and using 20 ppm. ......................................................................................... 104

4-VII: Execution times of PSO using ppm mass unit on HEK293 dataset files................................................................. 105

4-VIII: Comparison between PSO model and Param-Medic ............................................ 106

5-I: MS/MS spectra and downloaded information.......................................................... 119

5-II: The datasets used in this study................................................................................. 121
List of Tables—Continued

6-I: Search engines parameter values. ................................................................. 140
6-II: The identified and observed spectra and the unanimous process. ............... 140
6-III: Comparisons of the accuracy values of different tools and datasets. ........... 142
LIST OF FIGURES

1.1: The major steps of proteins identification from mass spectra. ........................ 2

1.2: General SEQUEST-style in searching observed spectra. ................................. 4

1.3: Times of scoring functions compared to the total execution times in X!Tandem and pFind search engines. ......................................................... 12

2.1: The major steps of protein identification from mass spectra. ........................ 34

2.2: General steps in indexing Target-Decoy databases in Crux-Tide ..................... 38

2.3: Phase 1 of Tide’s Indexing – Cleaving Proteins into Peptides ......................... 38

2.4: Phase 2 – Decoy Generation ............................................................................. 39

2.5: Phase 3: Decoy Protein Construction ............................................................... 41

2.6: Parallelizing Phase 1 ...................................................................................... 45

2.7: Parallel generator for decoy peptides and decoy FASTA ............................... 46

2.8: The new structures and the matrix of hashed peptides. ................................. 47

2.9: The sequential execution times and the percentage difference of original version and our improved version. ....................................................... 51

2.10: Memory consumption and the percentage difference between the original version and our improved version. .............................................. 52

2.11: Serial and Parallel Execution Times ............................................................ 53

2.12: Comparison between the execution times of parallel versions using 4 threads and 8 threads. ................................................................. 54

2.13: Speedups. (Speedup = Serial Time / Parallel Time) ...................................... 54

3.1: The performance of machine learning algorithms using Human01-pFind dataset. .......................................................... 71
List of Figures—Continued

3.2: The performance of machine learning algorithms using Human01-Comet dataset. ................................................................. 72
3.3: The performance of machine learning algorithms using Human06-pFind dataset. ................................................................. 72
3.4: The performance of machine learning algorithms using Human06-Comet dataset. ................................................................. 73
3.5: The performance of machine learning algorithms using Human05-Comet dataset. ................................................................. 74
3.6: The performance of machine learning algorithms using Mouse-pFind dataset. ................................................................. 74
3.7: The performance of machine learning algorithms using Mouse-Comet dataset. ................................................................. 75
3.8: The performance of machine learning algorithms using Soybean-pFind dataset. ................................................................. 75
3.9: The performance of machine learning algorithms using Soybean-Comet dataset. ................................................................. 76
3.10: The performance of machine learning algorithms using Rat-Comet dataset. ................................................................. 76
3.11: The performance of machine learning algorithms using HEK293-Comet dataset. ................................................................. 84
3.12: Average performance of various machine learning algorithms across the first 10 datasets in our experiment. ......................... 86
3.13: Average performance of various machine learning algorithms across HEK293 datasets in our experiment. ......................... 87
4.1: General SEQUEST-style in searching observed spectra. ...................... 93
4.2: The schematic of our PSO model. ................................................................. 96
4.3: Execution times of PSO using different mass units. ......................... 103
5.1: A naive design of distributed computing solution for the majority voting mass spectra interpretation system. ......................... 115
5.2: Machine learning models evaluation on Human01 Dataset. ...................... 123
5.3: Machine learning models evaluation on Human06 Dataset. ..................... 124
5.4: Machine learning models evaluation on Rat Dataset. ............................. 124
5.5: Machine learning models evaluation on Soybean Dataset. ...................... 125
5.6: Machine learning models evaluation on HEK293 Dataset ...................... 129

6.1: Major components of a naive distributed computing solution design for the majority voting mass spectra interpretation system. ............................................................................................................... 137
6.2: The distribution of all observed spectra and the unanimous ones based on the precursor values. Yeast (left) and Human (right). .......................................................................................................... 141
6.3: Peaks retained after applying uniform cut-off. ........................................ 142
6.4: Comparison of accuracy values of the four search engines results after applying Top X% on Yeast spectra ................................................................. 144
6.5: Comparison of accuracy values of the four search engines results after applying Top X% on human spectra ................................................................. 144
6.6: Retained peaks after applying Top 10% on Yeast and Human spectra. ................................................................................................................................. 145
6.7: Comparison of accuracy values of the four search engines results after applying multistage reduction on Yeast spectra. ............................. 145
6.8: Comparison of accuracy values of the four search engines results after applying multistage reduction on Human spectra. ...................... 146
6.9: Retained peaks after applying Multistage and Top 10% on Yeast and Human spectra. ............................................................................................................. 146
6.10: Comparison of accuracy values of the four search engines results after applying random sampling on Yeast spectra. ...................... 147
6.11: A comparison of accuracy values of the four search engines results after applying random sampling on Human spectra. ...................... 147
6.12: A comparison of accuracy values of Comet and Crux-Tide search engines results after applying Top X% sampling on HEK293 spectra. .................................................................................................................. 152
CHAPTER 1

INTRODUCTION

1.1 Mass Spectrometry-based Proteomics

Mass spectrometry based proteomics is one of the most popular techniques in protein identification. In terms of the number of publications, Google scholar shows that there are 12,300 publications since the year 2000 in which the keywords "mass spectrometry proteomics", appear together. 70% of those papers were published since 2010, and 23% since 2015. This indicates the heavy dependency and usage of such process in research fields such as: computational biology, medical sciences, etc.

In mass spectrometry based proteomics, the identification of peptides, the basic blocks of proteins with a smaller size of linked amino acids, is a fundamental task in protein identifications/characterization procedures. There is no single, straight forward step to detect amino acids sequence directly from the biological samples. The protein is usually identified by grouping the matched peptides of the input set of spectra. In general, there are three basic techniques to identify/match the peptides from mass spectra: (1) De novo methods [1] where the amino acids are identified based on direct calculations on the spectrum’s peaks, (2) Spectral library approach in which each input spectrum is searched against previously identified spectra [2], and the most common one is (3) Searching the input spectra against proteins’ database [3, 4].

Figure 1 shows major steps of protein identification starting from the biological samples where the unknown proteins are digested using enzymes (Trypsin for example) into peptides. Those peptides are fed into high throughput mass spectrometers where a massive amount of
spectra is generated in a short time. For instance, more than 24 gigabytes of compressed spectra are produced from a thermo fusion device in one day [5]. A Quadrupole-Orbitrap instrument generates around 14 Gigabytes of raw spectra per day [6]. One gigabyte of raw spectra is generated per hour by a Q-T instrument and LC MS/MS [7]. After that, the computational challenge starts in order to identify the best matches of spectra by De novo methods, spectral library or searching another massive size of protein database, and finally grouping the peptides into proteins. Some solutions could combine different major techniques together, for example De novo and spectral library.

1.2 Searching Against Proteins’ Database

Most of the current search engines are still inspired by the SEQUEST approach in observed spectra interpretation. Figure 2 shows the major tasks and components of a typical observed spectra search engine. One of the main computational challenges in protein identification and related problems is the data size.

Figure 1.1: The major steps of proteins identification from mass spectra.
The advance technology of spectrometers introduces a major challenge for current search engines, where the volume of generated spectra could easily reach terabytes [8]. Obviously, as figure 2 depicts, not only the size of the observed spectra is the only challenge, but also the database size, which is increasingly growing [9]. The larger the database, the larger the number of candidates and hence the more execution time and computation resources are needed. Of course, other parameters could also impact the performance, such as the window size (w) of the candidate peptides’ precursor values [10]. More comparisons are needed as the value of (w) increases, thus requiring more computing resources.

Candidate peptides are those peptides which have precursor values equal to the observed spectrum precursor value ± W, where W is a user defined mass window tolerance. For example, given W=3, an experimental spectrum of precursor mass value of 700 will be compared against all theoretical peptides of masses belong to [697, 703], inclusive. The number of theoretical retrieved peptides from the database, for each observed spectrum, mainly depends on three main factors: the precursor value of the observed spectrum, the database size, and the window tolerance value.

Table 1-I shows how the number of candidates differs according to the different values of the aforementioned factors in Crux-Tide search engine. To draw the results of this table, two observed spectra with different precursor values were randomly chosen; namely precursor values 1271.67, and 1489.53, respectively. The computation of searching MS/MS spectra is mainly dominated by those three variables. Those parameters are very important to be taken into account when any high throughput solution is being evaluated.
Table 1-I: The number of candidates for each spectrum based on three different variables.

<table>
<thead>
<tr>
<th>Window tolerance value = 3</th>
<th>Database 1</th>
<th>Database 2</th>
<th>Database 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed Spectra precursor value = 1271.67</td>
<td>17,213 Peptides</td>
<td>42,804 Peptides</td>
<td>93,300 Peptides</td>
</tr>
<tr>
<td>Observed Spectra precursor value = 1489.53</td>
<td>15,271 Peptides</td>
<td>37,417 Peptides</td>
<td>80,571 Peptides</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Window tolerance value = 4</th>
<th>Database 1</th>
<th>Database 2</th>
<th>Database 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed Spectra precursor value = 1271.67</td>
<td>22,909 Peptides</td>
<td>56,930 Peptides</td>
<td>124,120 Peptides</td>
</tr>
<tr>
<td>Observed Spectra precursor value = 1489.53</td>
<td>20,173 Peptides</td>
<td>49,543 Peptides</td>
<td>106,965 Peptides</td>
</tr>
</tbody>
</table>

1.3 Proteins Database Search Engines

Googling about proteins search engines results in many hits. These could be mainly categorized as: commercial such as SEQUEST [11], MASCOT [12], PEAKS [13, 14], freeware or open source, such as Crux [15], OMSSA [16], X!TANDEM [17]. List of search engines and protein identification tools with concise description can be found at wiki webpage [18].

The experiments conducted to build this dissertation’s research study mainly depend on five search engines; namely, Crux-Tide [19], pFind [20], Comet [21], X!Tandem and OMSSA.
Crux-Tide shows dramatic speedup in searching observed spectra against a set of candidate theoretical peptides retrieved from indexed peptides database. Crux-Tide utilizes cross correlation in order to rank candidate peptides, which the input observed spectrum possibly represents. The top candidates are then filtered by percolator [22], which reports the best match given the inputs. Crux-Tide engine is the accelerated version of the SEQUEST search engine where several smart optimizations are implemented.

pFind tool depends on an advanced approach to strengthen the dot product scores by utilizing the kernel trick, namely of including the relative information among spectrum fragments ions. The usage of kernel trick reduced the error rate of peptide-to-spectrum-match (PSM) by around 10% compared to SEQUEST and Sonar MS/MS results [23].

Comet is a multithreading search engine which supports multiple types of inputs. It can search the MSMS spectra where the indexing phase on the FASTA files (Proteins’ databases) can be skipped. X!Tandem is a popular search engine which depends on hypergeometric score in matching observed spectra against the theoretical ones, where the score, namely hyperscore, includes the number of y- and b-ions as incorporated in the equation:

\[ \text{HyperScore} = N_b! \times N_y! \times \sum_{i=0}^{n} I_i \times P_i \]

Where \( N_b \) and \( N_y \) are the number of b- and y-ions, respectively. \( I_i \) is the spectrum i intensities, \( P_i \in \{0, 1\} \) and \( n \) is number of peaks in the observed spectra. The preliminary score of X!Tandem is a dot product of the peaks of the observed and theoretical candidate spectra. Because only the shared or similar peaks are considered, this is equivalent to the sum of the intensities of the matched y and b ions. The values of Hyperscores are followed by empirical calculations of E-value that measure how good each match is for each observed spectrum. Given
a hyperscore $H$, E-value is the average number of peptides expected to have a Hyperscore value, $h$, where $h \geq H$, and $H$ is chosen randomly. Therefore, the larger the value of $E$, the more likely the match is correct [24]. While calculating the E-values, X!Tandem depends on empirical study, where a histogram of scores is used for each observed spectrum for a statistical distribution fitting.

OMSSA depends on Poisson probability distribution to calculate the E-values for each spectrum scores which were calculated based on the number of matches of shared peaks:

$$P(X = x) = \frac{\mu^x}{x!} e^{-\mu}$$

Where $\mu$ is estimated as a function of precursor ion, charge, number of peaks of the observed and number of theoretical m/z values of all candidates peptides for that observed spectrum [24].

The reasons behind selecting the above search engines to conduct the experiments are mainly the diversity of their methods in matching and evaluating the peptides-to-spectrum matches, or PSMs and their availability for researchers. Querying the databases through different search engines could add value to results of some experiments, for example those experiments of building machine learning solutions or evaluating sampling techniques, in the sense that the results can be generalizable.

1.4 The Choice of Databases in Proteomics

The choice of protein sequences databases has a major impact on the final results of proteomics analysis as well as on the required computational resources. In fact, regardless of the available computational resources, the choice of database(s) depends mainly on the objective of the experiment or research.
1.4.1 The Protein Databases for Single Organism Experiments: the Load of Extra Sequences

One of the limitations of any protein database search engine, compared to de novo sequencing, is the dependency of analysis on the input database. All identifications should map the input spectra to peptides that already exist in the database. Therefore, the inclusion of all possible sample proteins in the protein database is an important factor of the searching process. Possible non-identified spectra may represent novel peptides/proteins that do not exist in the database or incorrectly matches to different peptides/proteins in the database (i.e. false positive matches) [25, 26].

Depending on the ultimate target of a proteomics research project, the protein databases could consist of the original set of pre-identified protein sequences (e.g. for human datasets,ipi.Human v.3.87) or could be enriched by artificial sequences. For instance, in order to reveal all possible novel peptides/proteins in the input sample, artificial sequences may be generated using six-frame translation of an organism genome. Additionally, researchers may take into account the variations of protein sequences as a result of nucleotide polymorphisms (i.e. single nucleotide polymorphisms, SNPs). These SNPs may be related to specific conditions, including disease biomarkers [27, 28]. A major concern in these approaches is that the size of the organism database may be increased from a few megabytes into gigabytes. For example, from ~ 45 MB into ~3.2 GB when the human genome (UCSC V1.9) is directly translated using six-frame translation [26]. Indeed, using such large databases present two main challenges: the introduced high false positive rates and the computational resources needed to search and index the database, especially using Target/Decoy strategy.

Excluding short and/or redundant sequences of the aforementioned large databases could be one of the possible solutions to search-space reduction. However, more powerful and efficient
computational algorithms are still needed to handle larger databases than those already used in single organism proteomics. In the following subsection the need for searching against larger databases in metaproteomics experiments is exhibited, with databases of multiple organisms involved in the searching process.

1.4.2 Large Protein Databases in Metaproteomics

While several proteomics projects study proteins from one species, the metaproteomics field of research focuses on studies of proteins from multiple organisms. The general steps of the workflow to identify MS/MS spectra in both of them are the same. The core function is to map the input experimental spectra to peptides retrieved from the input database. However, a main difference or challenge in metaproteomics experiments is data size; both the experimental spectra and the database [29].

Metaproteomics is an important tool to study and analyze complex ecosystems. It mainly focuses on microorganisms’ properties, functions and their relationships in complex ecosystems. Since identification of proteins is essential in understanding many of the functions in organisms’ cells, incompleteness of protein databases of some microorganisms is a common challenge in peptide identification. Additionally, there is a large number of species that contribute to the metaproteoms, making the required databases to identify proteins large and complex. Consequently, computational resources and powerful algorithms are crucial components of any metaproteomics experiment. For example, using the UniProt/TrEMBL database, which contains protein sequences of only 788,214 species (based on the status on 01/16/2018, https://www.ebi.ac.uk/uniprot/TrEMBLstats), or the entire NCBI reference database, could result in very poor performance or system failure due to limitations in hardware and the current protein
search engines [29]. Since the number of microbial species on the earth could reach a trillion [30], efficient searching of possible spectra is a challenge for protein search engine developers and software engineers while designing high throughput software.

Metaproteomics is considered a main tool which includes large-scale analysis of proteins in micro and complex communities. For example, large-scale experiments have been conducted using complex samples of plants and soil [31]. Metaproteomics studies also have a wide range of applications including human healthcare, livestock, and food production [32]. For example, in human healthcare, a recent analysis of fecal samples of 15 patients with cystic fibrosis was conducted, and results showed that metaproteomics is a powerful tool in identifying disease biomarkers. These experiments were conducted using X!Tandem search engine (version CYCLONE 2010) on the NCBI database (downloaded in August 2013) which has 31,351,517 sequences [33]. In another interesting application, metaproteomics was used to study food microbiota at the molecular level. Metaproteomics was also demonstrated as an effective tool for monitoring safety and quality of Italian cheese. These experiments were conducted using MASCOT (v 2.4) search engine on NCBInr bacterial database; 11,349,194 sequences [34].

Cloud-based solutions, real-time processing of MSMS data, scalable solutions of database searching over distributed environments, and inclusion of powerful computational resources such as GPUs (Graphical processing units) are some suggested solutions to effectively search the increased amount of data in proteomics/metaproteomics projects [29]. The large sizes of protein databases as well as the experimental spectra require the proteomics research community to revisit the design of currently available protein search engines. In order to search large databases such as those utilized in the above experiments, the typical workstations could be insufficient, especially with the sequential computational software. To take advantage of all the
available computational resources in a workstation or computing cluster, parallel processing may be one of the efficient solutions. Machine learning algorithms such as MS/MS filters could also be useful in managing the load of the huge amounts of data getting processed in the system.

In order to efficiently use all the available computational cores, the searching phase in protein search engines could easily be the most important candidate to be run in parallel. However, in some search engines like Crux-Tide, protein database indexing is also an important phase that needs optimization to have complete and efficient solutions. That is because of the large sizes of protein databases, especially in metaproteomics studies, and the frequent updates on some protein databases such as UniProtKB, which is updated monthly [25]. In chapter II of this dissertation, a parallel solution to indexing target/decoy protein databases is discussed to boost the performance of one of the fastest search engines currently available as open source.

1.5 Computational Challenges from the Size of Observed Spectra in Proteomics

Huge amount of raw spectra introduces several challenges into any proteomics computation, either on one machine or clusters of several machines. Those systems should use efficient techniques to cope with several of the following challenges, but are not limited to, or at least to mitigate their impact on the whole process:

1.5.1 Computation Speed and Throughput

Peptide deduction from observed spectrum using either database searching or De Novo inference depends mainly on the spectra peaks. Generally speaking, in matching process, a function or a set of functions are utilized to measure the matching quality, or simply the similarity, between the experimental spectrum and a theoretical candidate spectrum. Formally,
given an experimental spectrum $S$, and scoring function $F$, find the peptide $P$ from the set of candidate peptides which maximize the value of $F(S, P)$ [35]. There are several scoring functions discussed in the literature and currently used in protein identification tools. Some search tools such as X!Tandem, OMSSA, SEQUEST, MASCOT, depend on correlation-based algorithms starting from the simple dot product to advance cross correlation. Other tools depend on different techniques like empirically observed rules used in SpectrumMill, and statistically derived fragmentation frequencies used in PHENYX. [2]. Regardless of the utilized similarity technique, the spectra peaks are the main source of complexity in similarity computation. The larger the number of peaks, the more execution time needed. Given the billions of observed spectra, and other billions of theoretical candidates [10, 36, 37], the execution time is definitely not cheap. De-noising strategies and peaks reductions could help the search engines by reducing the number of peaks which, in turn, reduces the execution time. Figure 3 shows the scoring function time compared to the total execution in two popular search engines; X!Tandem and pFind [38].

Analyzing very large volumes of data is considered risky due to long execution times which introduce probable failures given the resource constraints and the computation models; serial or parallel. Processes compete over limited resources. On the other hand, processing batch data could be more flexible in optimizing the processes and the required sizes, but in-stream processing is extremely challenging and the spectra size must be as small as possible, but yet useful.
1.5.2 Memory Footprint

Transferring the original spectra from their source into memory system will add more overhead on the required memory space, and the system swap space. Despite the fast internal data buses between the computation system units, the deluge of many gigabytes of raw spectra for processing places a heavy burden on the memory system.

1.5.3 File Transfer

Broadly speaking, any simple experiment in transferring massive size of data could significantly slow the performance of even local area networks in handling this cumbersome task [39-41]. Establishing a mechanism to reduce the impact of size in transferring process is one of the hot big data research areas. Conventional methods like HTTP and FTP are inefficient in moving this sheer amount of observed spectra and the required databases. Moreover, looking for initiative and modern shared storage options for proteomics requires high speed transferring solutions, and those solutions could be new network technology or protocols [42-43], or smart data reductions or compressions [44]. After that, we could swiftly download /upload (data sharing) spectra files from/to spectra repositories such as the Universal Protein Resource,
http://www.uniprot.org, and http://www.peptideatlas.org/repository/ for academic research and lab experiments.

1.5.4 The Flood of Spectra in HPC Solutions

Dividing the spectra load among threads or processes is the straightforward solution where each thread/process executes the same code over different data (SIMD or SPMD), for example [45]. Whether the spectra are read serially or in parallel from their source into the HPC system, the factor of spectrum size should be taken into account while moving the spectra over the system computation resources. In shared memory HPC for proteomics, the storage bottleneck is one of the big limitations [46], and the storage latencies should be involved in the performance formula [47], therefore, reducing the spectra size could be part of the solution. Some proteomics HPC solution depends on distributed computing in order to utilize larger number of processes and their own memories [48-52]. The spectra are transferred to the processes either as one unit before the processing (matching) phases or batches to manage the load balance issue [10, 49]. In both cases, the speed of transferring the spectra is one significant operand in the equation of computation speed. Given the billions of spectra, and the same interconnection network configurations and conditions, the light weight spectrum adds advantage to the whole system performance, since the time to distribute spectra of average size S is larger than the time needed to distribute spectra of average size S/c, where c could be any integer larger than 1.

1.5.5 Storage and Backup

Despite the availability of cheap secondary storage and cloud storage, generating massive amount of raw spectra from several spectrometers will need several (tens if not hundreds or even
thousands) disks after \( \Delta t \) of time. Maintaining this hill of data for future processing and analysis or as public repository [53 - 55] (classified ones are listed in [56]), then shrinking the size to avoid the need of larger number of disks and mirror disks could add significant value in terms of storage cost and space management. Despite the various challenges in building big data proteomics public repositories, some solutions like PeptideAtlas, PRIDE, GAPP, etc [57-59] could be improved and utilized to cope with the current flood of mass spectrometry data. More collaboration in terms of sharing data and tools becomes an urgent goal for countries or research organizations to help advance proteogenomics.

1.5.6 Applications of Machine Learning

Clustering and classification are typically used in different fields of research. Observed spectra are clustered in order to isolate similar spectra or classified to filter low quality spectra and thus reduce the computation, communication and resources [60-62]. Clustering and classification algorithms could also be affected by the number of input spectra and the peaks in each spectrum. As alternatives to the serial traditional machine learning algorithms, fast implementation or available parallel algorithms for clustering and classification, for example [63-65], could support building machine learning models. However, the memory system should be able to handle the incoming spectra from the disk or flood directly from spectrometers.

1.6 Major Issues in the Current MS/MS Search Engines

Improving the accuracy along with the performance of protein identification is the main goal of the proteomics research. However, up to the time of writing this dissertation, the need for high throughput search engines of high accuracy exists. Despite the tens of available search
engines, the field of proteomics still needs more efficient engines to cope with the huge amount of MS/MS data, besides the challenge of continuously growing protein databases’ size. This section briefly discusses few major limitations and challenges in the current search engines, mainly: the speed of indexing proteins’ databases, searching throughput, searching parameters optimization and the reliability of the Peptide-to-Spectrum Matches (PSMs).

1.6.1 Target-Deoy Databases Indexing

Indexing the peptides’ database based on their precursor masses or their fragment masses as in [66] provides more efficient retrieving since the candidates are swiftly located instead of searching the whole database, and that dramatically reduces the searching time of the original SEQUEST style where full database scan was performed for each new experimental spectrum.

Target-Deoy database is currently the method of choice to assess the quality of Proteins’ search engines. Decoy versions of real peptides are generated and injected to the same database of real ones with different labels, or indexed into different database. Quality of search engines results is assessed based on the number of decoys retrieved as hits. Indexing protein database is considered an infrequent phase in database search engines, while the most emphasis is on the searching phase which should be fast enough to efficiently cope with the massive experimental data generated by mass spectrometers. This could be one of the reasons behind so many proposed parallel solutions for database searching phase. However, indexing and decoy generation is also an essential task in the whole process of protein identification and it should be done efficiently.

Changing the digestion enzyme, new protein sequences, or generating more decoys for quality experiments could easily result in rebuilding the whole database. Since the protein
database is increasingly growing and large databases are usually used in metaproteomics projects (see section 1.4), a fast indexing/re-indexing solution could add significant value in database search engines development and performance. In Crux-Tide, one of the fastest search engines available as open source, it takes unreasonable times and memory spaces and still has a room of improvement. The unreasonable memory spaces used while indexing FASTA files makes this system cripple and could cause failure, system crash, or very poor performance especially if the system memory gets overflow and virtual memory becomes in use. For instance, using the author’s machine, a workstation of 16 Xeon cores and 128 GB of RAM, even with relatively small to medium sizes, indexing 1 GB and 3 GB of protein sequences consumes around 26 GB and 977 seconds, 100 GB of RAM and around 4300 seconds, respectively.

1.6.2 Searching Massive MS/MS against another Massive Databases

Despite the linear relation between the number of observed spectra and the searching time, the current protein search engines, even the parallel versions, could take several hours to search a large amount of MSMS spectra, which can be generated in a short time. After a laborious searching process, some (and at times, majority) of the observed spectra are labeled as non-identifiable. The computation load and complexity in MS/MS searching process depend on several influential factors, including but not limited to: the database size, the tolerance window, the precursor value of the observed spectrum, and the number of observed spectra. In terms of database’s size, the typical size of protein databases is rapidly increasing [37]. The larger the database, the more candidates will be possibly retrieved for each observed spectrum. Based on the precursor value of the observed spectrum, the matching could be retrieved from few of the candidates up to millions of them. The advancing technology in mass spectrometry devices adds
new challenges to the process of protein identification. Newer devices are able to generate Gigabytes of spectra in a very short time, which leads to more extensive computing in addition to more precise and accurate peptide identification.

Peptide identification from these extensive datasets is a typical Big data problem that handicaps the current protein search engines; these engines struggle to handle even small to medium sized observed spectra and databases on a typical workstation. In addition to the size challenge, the current search engines have different abilities in identifying each input spectrum. In other words, coverage is different from one search engine to another for the same input of both spectra and database. Not all input spectra or peaks within observed spectra include important information about the proteins in the original biological sample. Therefore, efficient solutions to exclude “possibly irrelevant” spectra can decrease searching time, and also improve the accuracy of identification.

1.6.3 Searching Optimization

The PSMs in protein identification is controlled by several user-defined parameters. It does not necessarily that parameters values used by users will guarantee the optimal setting. That is because the inputs could have different properties where a unified parameters setting is not always ending up by the desired output. The database size, the precursor value and the window tolerance (w) affect both the load of computations and the number of possible correct peptide-spectrum matches (PSMs). Since the results of scoring functions are the main input of the post-processing phase, in which matches are filtered, those parameters have a direct effect on the final PSMs. In fact, increasing/changing any of these three parameters could result in a much larger set of candidates for the matching process of an observed spectrum, thereby making the post-
processing phase computationally intensive. Randomly setting these parameters could result in very poor searching speed and results’ quality as well. Therefore, optimization of these parameters will improve a search engine’s capability and directly impact the quality of results.

The peptide tolerance window parameter \( w \) is usually set as a constant value in most of the currently available search engines. Current search engines depend on a rule of thumb value, for example, in most of them, including Comet and Crux-Tide, the default value is 3 Da.

1.6.4 The Reliability of the PSMs and the Overlap among Multiple Search Engines

Beside the challenge of data size where the capacity of the current search engines is way beyond the currently generated hills of mass spectrometry [67], users could also get confused because of the number and the diversity of developed and available search engines, particularly, the different peptide identifications they report for the same inputs either in the deduction of peptides or the number of identifications [68]. These low intersection ratios could mislead the biologists in protein identification since each one could report different proteins. Clearly, it is hard to answer the question -- which tool should a laboratory depend on and rely upon -- since accuracy of peptide deduction itself is the main quest. As test cases, without loss in generality, we show results from three search engine in Tables 1-II using HEK293 dataset and four different ones in Tables 1-III (using different dataset), similar behavior can easily occur with different tools, or may be even much worse. The searching parameters have been unified except one different parameter for Comet, which is the \( \text{fragment}_\text{bin}_\text{tol} = 0.02 \). This is based on the developers of Comet recommendations for high resolution datasets. The overlap ratios are calculated using Jaccard index which is the intersection over union.
HEK293 has 24 peaks files that consist of 1,119,064 spectra in total (based on Comet v 3.1). Using human protein sequences database (ipi.Human v3.87) and tolerance window of 20 ppm; Comet was able to identify 34% of the spectra. pFind identified only 23% of them, and Crux-Tide just 20%.

Barring the arguments for or against the number or the choice of tools, this discrepancy introduces a controversial issue related to these different tools for peptide deduction and protein identification -- why should we trust or distrust the results of any tool? The intersection ratios also show how hard it is for one to predict from other tools’ answers given an answer from one of them. The prediction in this case could help in identifying one tool or a subset of them to run on behalf of the others, which certainly implies less execution time and efficient use of resources.

Table 1-II: Overlap ratios (Jaccard index) among the search engines using HEK293 Dataset.

| Search Engines | Overlap Ratio = $\frac{|Results\ of\ SE1 \cap Results\ of\ SE2|}{|Results\ of\ SE1 \cup Results\ of\ SE2|}$ |
|----------------|-----------------------------------------------|
| Comet          | pFind                                         |
| Comet          | Crux-Tide                                     |
| Crux-Tide      | pFind                                         |
|                |                                               | 61%                                      |
|                |                                               | 57%                                      |
|                |                                               | 47%                                      |

Table 1-III: Overlap ratios among the search engines using Human Dataset.

| Search Engines | Overlap Ratio = $\frac{|Results\ of\ SE1 \cap Results\ of\ SE2|}{|Results\ of\ SE1 \cup Results\ of\ SE2|}$ |
|----------------|-----------------------------------------------|
| Crux-Tide      | pFind                                         |
| Crux-Tide      | OMSSA                                         |
| Crux-Tide      | X!Tandem                                      |
| X!Tandem        | pFind                                         |
| X!Tandem        | OMSSA                                         |
| OMSSA           | pFind                                         |
|                |                                               | 55%                                      |
|                |                                               | 54%                                      |
|                |                                               | 55%                                      |
|                |                                               | 91%                                      |
|                |                                               | 88%                                      |
|                |                                               | 91%                                      |
Table 1-III shows higher overlap ratios among the search engines, but nonetheless different tools identify different peptides with a reasonably high percentage for the same observe spectra. The ratios of identification, or coverage in different words, are very low; Crux-Tide search engine reported around 19% of the input spectra, whereas the others reported only around 13% of them.

Answering the question about the tools of choice warrants deep analysis and experiments to establish, at least, some rules of thumb. Obviously, biologists look for the coverage (the percentage of identified spectra), the fastest tool, and of course the accuracy. Therefore, one solution could be a combination of different tools [69-71]. Consensus-based proteomics, where the spectrum is identified by consensus scoring or voting, could be used as well to increase the degree of trustworthiness and the reliability in peptides’ deduction results [72-73]. Another alternative is the majority voting systems [74].

1.7 Dissertation’s Contributions

This dissertation provides and evaluates solutions to the discussed limitations and challenges of Section 1.5. Subsections below show the problem(s), methods and the major results contained in each chapter of the dissertation.

1.7.1 Chapter 2: A Multithreading and Hashing Technique for Indexing Target-Decoy Peptides Databases.

This chapter shows an analysis of the serial algorithm of the Target-Decoy indexing phase in Crux-Tide search engine in details with improvement possibilities. It also describes a parallel shared memory solution using OpenMP.
To completely break up the dependency in the serial algorithms, a clever hashing technique is utilized to localize the process. The parallel solution and the hashing technique together are able to reduce the computation cost by approximately 70-80% using few threads. Besides the parallelization, we redesign part of the serial code so that the memory consumption becomes more efficient. The parallel version can index the same files using around two-thirds of the memory space that the serial version consumes. This solution could impact and support future distributed developments of Crux-Tide searching phase, where each parallel unit could rank the observed spectra independently.

1.7.2 Chapter 3: Deep vs. Shallow Learning-based Filters of MS/MS Spectra in Support of Protein Search Engines

This chapter shows the role of machine learning in building an efficient MSMS filter to remove non-identifiable spectra. Deep learning algorithm is compared and evaluated using 9 shallow learning algorithms with different configurations. Using 11 different datasets of more than a million of spectra generated from two different search engines, different instruments, different sizes and from different species, we observe that deep learning models are powerful in filtering MSMS spectra based on the experiments results. The developed deep learning model can exclude around 45%-50% of the non-identifiable spectra while losing, on average, only 9%-12% of the identifiable ones. As for shallow learning, algorithms of: Random Forest, Support Vector Machine and Neural Networks showed encouraging results, eliminating, on average, 60%-70% of the non-identifiable spectra but also loose a higher percentage around 25%-35% of the identifiable spectra. Deep learning algorithms may be more useful especially in instances where the protein(s) of interest are in lower cellular or tissue concentration, while the other algorithms may be more useful for concentrated or more highly expressed proteins.
Chapter 4: Optimizing Protein Search Engines using Particle Swarm Optimization

In this chapter, to maximize PSMs, we evaluate and present a particle swarm optimization (PSO) to improve the coverage of search engines by picking the optimal value for the tolerance window parameter. This influential parameter has an important role in the final number of PSMs. The results show that this biologically-inspired algorithm can be utilized to find peptide mass window tolerance values that facilitate, for example, Comet search engine to increase peptide spectra matches, resulting in improved peptide identification. As an important lesson based on these experiments, the results also show experimental evidence that an open search (i.e., wide tolerance window) does not always optimize spectra matching using the current search engines and that narrow tolerance windows improve the coverage of protein search engines.

For performance evaluation, the coverage of the Comet search engine was studied using the mass tolerance value recommended by our PSO model, the recently developed Param-Medic tool [75] and the rule of thumb value (i.e. the default value 3 Da). For a fair comparison, two experiments with different datasets and different sizes were conducted on the same machines. First, nine different MS/MS datasets generated by different instruments were randomly selected from the PRIDE public repository. In comparison to Param-Medic and the default rule of thumb values, results showed that the PSO increases the number of correct PSMs. Furthermore, PSO is able to handle all of the datasets and find the best tolerance window value, which when used, resulted in improved number of Comet-identified PSMs. Results also showed that given the current filtering approaches, narrower mass tolerance windows sometimes improve search engine coverage. In the second experiment Param-Medic, unfortunately, cannot find enough information to suggest the best tolerance window to search HEK293 MS/MS spectra. The PSO
results, therefore, were compared to the results of using a fixed value of tolerance window (i.e. 20 ppm).

1.7.4 Chapter 5: Deep Learning-based MSMS Spectra Reduction in Support of Running Multiple Protein Search Engines on Cloud

The main target of this chapter and the next one (chapter 6) is to support running multiple search engines by reducing the required data, or in different words, to exclude the unnecessary computation load. A deep learning model is developed and evaluated in order to mitigate the traffic over cloud network and, thus reduce the cost of cloud computing. The model supports distributed majority voting solutions, over cloud or clusters. The model, which depends on the top 50 intensities and their m/z (mass-to-charge ratio) values of each spectrum, removes any spectrum which is predicted not to pass the majority voting of the participated search engines. The results using three search engines namely: pFind, Comet and X!Tandem, and four different datasets are promising and promote the investment in deep learning to solve such type of Big data problems.

1.7.5 Chapter 6: Towards Centralized MS/MS Spectra Preprocessing in Cloud-based Ensemble Solutions of Multiple Search Engines

In this chapter, different statistical reduction techniques of MS/MS peaks have been evaluated using four popular protein search engines. The main objective of these experiments is to build central preprocessing where only useful peaks are sent over the cloud or a distributed network. In order to fairly evaluate the results, a ground truth unanimous-based datasets were built for two different species; yeast and human. The results showed significant peak reduction, where only around 30% of the spectra peaks are enough to report reliable identifications from the
semantically different search engines used in this study. These results were also corroborated on large datasets. For example, for the dataset HEK293 of more than a million high resolution MS/MS spectra, the results also showed that 30%-40% of the peaks are enough for Comet and Crux-Tide search engines to report, on average, 98% of the same results as if they received the original peaks.
References


[38] Li Y., Chi H., Xia L., Chu X., Accelerating the scoring module of mass spectrometry-based peptide identification using GPUs. BMC Bioinformatics, 15 (2014) 121.


CHAPTER 2

A MULTITHREADING AND HASHING TECHNIQUE FOR INDEXING TARGET-DECOY PEPTIDES DATABASES

This chapter presents the contributions of two publications:

(1) A MULTITHREADING AND HASHING TECHNIQUE FOR INDEXING TARGET-DECOY PEPTIDES DATABASES. Concurrency and Computation: Practice and Experience. Accepted, 2017, and


2.1 Summary

Target-Decoy database is currently the method of choice to assess the quality of Proteins’ search engines. Decoy versions of real peptides are generated and injected to the same database of real ones with different labels. Quality of search engines results is assessed based on the number of decoys retrieved as hits. In Crux-Tide search engine, which is one of the fastest search engines currently available, the process of indexing and generating decoys is computationally expensive. In this paper, we analyze the serial algorithm in detail and show improvement possibilities, and then describe a parallel shared memory solution using OpenMP. To completely break up the dependency in the serial algorithms, a clever hashing technique is utilized to localize the process. The parallel solution and the hashing technique together are able to reduce
the computation cost by approximately 70-80% using few threads. Besides the parallelization, we redesign part of the serial code so that the memory consumption becomes more efficient. The parallel version can index the same files using around two-third of the memory space that the serial version consumes. This solution could impact and support future distributed developments of Crux-Tide searching phase, where each parallel unit could rank the observed spectra independently. The parallel solution shows its value more in metaproteomics studies, where large protein database is typically used in identifying proteins from complex communities and individual organisms cannot be identified in advance. Section 1.4 briefly discussed metaproteomics and some of its applications where very large databases of protein sequences are being used to study complex environments.

2.2 Introduction

Mass spectrometry-based proteomics is one of the most popular techniques in protein identification. The identification of peptides, the basic blocks of proteins, is a fundamental task in this technique. There is no single, straight forward step to detect protein’s amino acids sequence directly from the biological samples. The protein is usually deduced from the identified peptides of the input set of spectra. Figure 1 shows the major steps of protein identification starting from the biological samples where the unknown proteins are digested using enzymes, Trypsin for example, into peptides. Those peptides are fed into high throughput mass spectrometers where a massive amount of spectra is generated in a short time. In general, there are three basic techniques to identify the peptides from mass spectra: De novo method where the amino acids are identified based on direct calculations on the spectrum’s peaks, Spectral library
approach in which each input spectrum is searched against previously identified spectra, and the most common one is searching the input spectra against peptides’ database [1, 2]. In database searching, we can express it concisely in few main steps; (1) observed spectra are normalized and preprocessed, (2) scoring functions match and rank the observed spectra against the set of candidates peptides retrieved from the database. Those retrieved peptides fall within a precursor mass tolerance window (w) where the spectrum is expected to represent one of them. For example, for w = 3, the search engine matches the spectrum of precursor mass 700.00 against all peptides of precursor mass that belongs to [697.00, 703.00], inclusive. (3) Finally, matches are evaluated and the best ones are used in proteins deduction.

Peptides’ database search engines face several challenges to speed up the matching process. For instance, the cumbersome size of spectra generated by spectrometers [3, 4] is considered a burden on systems’ computation resources. The proteins’ database size, which is increasingly growing [5], is also a challenge which should be addressed towards building high throughput and scalable solutions. The larger the database, the larger the number of candidates that will be retrieved. This means usage of more computation resources. In addition to the size, efficient retrieving techniques have a significant impact on overall systems’ performance. Indexing the database based on the peptides masses is a major optimization step where the candidates are retrieved swiftly for the matching process, and this dramatically reduces the execution time compared to the original approach where the entire database is scanned for each input spectrum [6, 7].

On the other hand, measuring the quality of the scoring functions is another challenge. Before depending on their results, search engines should pass the quality assessment test which shows how much sensitive they are towards retrieving the correct matches. To assess the quality,
the Target-Decoy database (section 1.1) technique remains the common and dependable strategy currently used in proteins’ database search engines [8]. In terms of computation, Target-Decoy database has a drawback that it multiplies the size of cleaved peptides. New versions of peptides, or decoy peptides, are injected to assess the ability of search engines in separating and filtering out them. Beside the size, this technique adds another challenge to the indexing phase; the dependency in computations for decoys validation (section 2). Thus, smart and efficient techniques are needed to handle these indexing issues. In this paper we describe our parallel and hashing solution to improve the computation of Target-Decoy indexing.

2.2.1 Target-Decoy Database Strategy

In order to increase the matches’ quality and the trust worthiness degree in the scoring functions and their results, Target-Decoy provides a solution to estimate the false positive rate which is the number of decoys deduced with a score that exceeds a threshold. This strategy is
still the method of choice and it is applicable in most of the search engines. Constructing Target-Decoy databases is simple to implement. Either in one database or in two separate databases, decoy (fake) format of peptides are injected and indexed in the same way the indexer does with the target (real) ones. Decoys can be generated using several techniques [9]; they can be cleaved from the reverse order of proteins’ sequences. In this case, a similar cleavage method that is used with real sequences is utilized. In another technique, each cleaved target (real) peptide is replaced by its reversed or shuffled version. For example, a real peptide of sequence “NFLETVELQVGLK” could be represented by “NLFETVEQLGVLK” as its decoy. Clearly, the size of the database being searched will be double in size as the original one. This means that more disk space and main memory needed, besides more computation overhead in all phases; from indexing to reporting protein sequences.

This decoy strategy replaces other assessment techniques such as ion inspection [10], and post-processing matches filtering [11], and it is, currently, the most common method applied for matches’ assessment. However, it brings other challenges in terms of computation resources’ utilization. First, the indexer of Target-Decoy databases starts allocating more resources of both CPU and RAM. Second, the validation of generated decoys is computationally an expensive process, since the overlap between targets (i.e., real peptides) and decoys (i.e., fake peptides) must be kept as low as possible. This simply means checking the existence of each generated decoy by scanning all peptides in the database, and sometimes scanning the previously generated decoys as well in order not to generate duplicates decoys. The execution dependency slows down the computation and adds a challenge towards parallel solutions. In fact, the low degree of overlap is one of the assumptions that one should take care of when using target decoy database. For more details, reader could refer to [8].
Compared to the searching phase, indexing the database is an infrequent phase in search engines. Massive amount of observed spectra are generated in short time and the scoring function in searching phase should be able to efficiently rank them in a timely manner. For this purpose, several experiments were conducted to speed up the current search engines, see for example [12-17]. However, changing the cleavage method, the enzyme, enriching the database by more decoys for more quality experiment, or adding new protein sequences could easily result in rebuilding the whole database. Therefore, indexing and decoy generation is also an essential task and it should be accomplished efficiently, especially as the protein database is increasingly growing. Given the large sizes, faster indexer could add significant value in database search engines development and performance.

2.2.2 Target-Decoy Indexing in Crux-Tide

SEQUEST [18], the most popular database searching approach in identifying peptides from MS/MS spectra, is the inspirational approach for most of the currently used search engines. However, because of its slow performance, many modern tools implement SEQUEST along with improvements and optimizations. The matching speed is the core limitation that motivates the work of several faster search engines. For example; Crux [19], TurboSEQUEST, and Crux Tide [20] are some of the well-known toolkits that are inspired by SEQUEST style.

Crux-Tide shows dramatic speedup compared to other currently used search engines. Several optimizations are developed and injected into Crux-Tide, which makes the searching results to be returned faster in comparison with SEQUEST. More specifically, these optimizations include caching techniques, database indexing, efficient algorithms and data structures, see reference [6] for more details and discussions.
Indexing and preparing the Target-Decoy database in Crux-Tide is a very expensive process in terms of both CPU time and memory space. A simple profiling experiment on a reasonable specs workstation (namely, 16 dual Xeon 2.6 GHZ E 5 family processor and 124 GB of RAM), shows that indexing around 1 GB of FASTA file consumes more than 23 GB of RAM and around 30 minutes of CPU time. This simply indicates that this indexer will struggle if larger databases are needed to be indexed in Target-Decoy style.

In this paper, we extend our original work in speeding up the peptides’ database indexing phase [21] of Crux-Tide. We improve the previous work. Our new design eliminates some of the intermediate data structures and variables which directly improve the memory consumption. Our discussion later in section 6 shows the encouraging results towards building high throughput protein search engines.

2.3 Sequential Crux-Tide Indexer

Crux-Tide searches mass spectra against an indexed database which is one reason behind its speed. In the indexing phase, the raw sequences of proteins are fetched from FASTA file one-by-one into a cleavage process, where each protein sequence becomes a set of peptides (shorter sequences of amino acids). Once all targets are settled in their containers, the indexer generates and validates decoys. Finally, both of them (i.e., targets and decoys) are combined and indexed into one database. Figure 2 shows the basic steps of sequential Crux-Tide-Indexer.
As depicted in Figure 3, each fetched protein sequence is cleaved into their peptides and organized into three different containers along with their information (i.e., protein name, location within protein, etc.). The main container is a heap structure where the targets are sorted based on their masses. Another container will host only those unique and sorted peptides’ sequences. Their reference information will be hosted in the third container. This step could add some complexity to the indexer, but those containers are used, cleverly, to improve the processing in the next phases, especially in generating decoys.

Figure 4 shows the process of decoy generation in detail. For each target peptide fetched from the unique set a decoy will be generated either by reversing or shuffling the sequences. The default parameter of the decoy-format is “NC.” Other supported options are “C,” and “N.” These parameters are used to control the process of moving or shuffling the terminal amino acids while
generating decoys. For example, using “NC” option, the peptide “NFLETVELQVGLK” must be shuffled in a way that “N” and “K” are not moved from their original end-point positions. The following shuffles are valid, for instance, “NEFTVELQGVLLK”, or “NGVLLEFTVELQK”, while “NEFTVELQGVKL” is an invalid shuffle in “NC” peptide format. Using “C” format, the last letter must not be moved. For example, “EFTVENELQGVLLK” is valid, while “EFTVNELQGVKLL” is not. For the “N” shuffling option, the first letter must not be moved; for example, “NEFKTVELQGVLL” is valid, while “EFTVELQGVLLNK” is not. If the format is set to “NONE,” then the reverse order of amino acid letters is used to generate the decoy, which will be: “KLGQVLEVTLEFN”.

![Diagram](image.png)

Figure 2.4: Phase 2 – Decoy generation.

Generating valid decoys is the heart of this phase. In fact, it is the main source of complexity which makes this serial process requires some optimization and speedup. In this context, valid decoys are those fake peptides which are not those real ones cleaved from protein sequences and also not those that are generated as a decoy for other real peptide. For example, if a peptide sequence “ASQLR” is a real peptide, then generating a decoy of the same amino acids
letters in the same order (i.e., “ASQLR”) is an invalid decoy. As discussed before, the assumption behind Target-Decoy databases is to keep the overlap between targets and decoys as low as possible, or they should not overlap. A high degree of overlapping could destroy the essence of the Target-Decoy assessment, which will lead to biased results. In the serial version of Crux-Tide, this assumption creates a computational dependency which makes indexing large databases a kind of challenge, especially, in workstations that are usually available in biologists’ labs. Technically, each candidate decoy is searched against all previously generated targets and decoys. The comparisons are repeated until a valid form is generated or a threshold is reached. This looks as a very simple process to implement, but it is very expensive in terms of computations. Clearly, if a large database is needed to be indexed/re-indexed, serial Crux-Tide will be greedy towards only the memory, but, of course, not CPU cores. This could mean unreasonable time of execution. Successfully generated decoys, as seen in Figure 4, will be hosted into three different containers: a heap of real peptides; a set of sorted and unique decoys, and finally to a map that shows each target peptide and its decoy version.

Up to this point of processing, the combination of target and decoy peptides process is completed and they are ready to be written into the disk for the searching phase. In crux-Tide indexer, an extra step is necessary to generate decoy proteins. Similarly, those are fake versions of the whole protein sequences. This can be accomplished by two different ways in Crux-Tide; either reverses the original proteins’ sequences or replaces each peptide in the original, real, sequence by its decoy form. The former is easier and it can be done without any reference to the aforementioned data structures. On the other hand the later needs to scan some of them to retrieve the correct corresponding decoys, (Figure 5).
Given the above discussion, the worst case complexity to cleave $Pr$ proteins into $Pt$ peptides (on average), will be $O(Pr \times Pt \times (T_{insertIntoHeap} + T_{insertIntoSet} + T_{insertIntoMap}))$. The peptides’ lengths are assumed to be constant in this evaluation.

In the second phase, many read/write operations against sets and maps, sorted contents are required. Those containers maintain the content sorted and unique as in a set data structure in C++, for instance. In order to validate each generated decoy, the validation process searches and inserts many times into a map and sets of targets and decoys besides other containers. It can be easily seen that the worst case time complexity is $O(U \times (T_{ReadingFromMap} + T_{checkingExistence} + T_{inserting into heap} + T_{inserting into decoyset} + T_{inserting into targetdecoyMap} + T_{searching on target peptides’ set} + T_{searching on decoy peptides’ set}))$, where $U$ is the number of unique peptides. Clearly, this phase consumes more resources and adds a dependency challenge towards parallelization.

Regarding the time complexity of building decoy proteins, it can also be computed as: $O(Pr \times Pt \times (T_{searching to find the decoy version} + T_{replace} + T_{writing}))$. Note that, this evaluation is not based on protein sequences reversing technique. In case of reversing, the complexity will mainly depend on how many proteins exist in the FASTA file. In addition to the
above concise theoretical discussion for the three phases, one should highlight, in this case, the
fact that peptides’ databases usually maintain, possibly, billions of peptides. This results in a
penalty of memory and storage spaces and, of course, CPU execution time in preparation phases.
Another fact says that maintaining sorted and unique data structures could be a very efficient and
clever idea especially if the system frequently reads from them. On the other hand, inserting into
them is a bit costly, particularly if the size is very large.

In the next section, we will discuss our solution to improve the performance of Crux-Tide
indexer by using OpenMP multi-threading. Detailed experimental results are discussed in the
subsequent sections.

2.4 Multithreading Hash-based Solution

A naïve parallelizing idea is to simply divide the FASTA into smaller files and feed each
piece into a thread. Given that the same peptide could be a segment of thousands of different
proteins, unfortunately, this simple approach is impractical, and, moreover, the serial execution
could be better than the parallel version when we take into account the dependency among the
threads. As previously mentioned, each generated (or candidate) decoy should be validated by
searching containers of targets and decoys before they can be approved and added to the
approved decoys. In this case, we could have two choices, either to contact other threads to
validate the decoy, or ask them to send their containers and validate locally. However, both of
these could result in poor performance, even could be worse than serial, since millions or even
billions of peptides (or their references) need to be transferred among threads for validity
checking. Moving or sharing billions of them over the interconnection network is a deluge of
strings and can cause a significant overhead. Therefore, mitigating or avoiding the

42
interconnection, among threads or processes in distributed environments, can add a significant
value to any parallel solution.

Another challenge is the dependency between preparing the target peptides and their
decoys. The decoy generator cannot work and validate the decoys until all targets get generated.
That introduces an inevitable serial execution (i.e., phases’ sequential dependency).

In our proposed solution, we divide proteins among threads. A hashing technique is
utilized to break up the dependency among the parallel threads execution. Hashing techniques
isolate those comparable, in terms of decoy validation, targets and decoys peptides into the same
hash buckets. This localizes all comparisons and executes them against smaller size containers.
We use domain knowledge in order to hash them. Notice that the values of peptides’ molecular
masses are the same even if the order of amino acids letters is changed. Then generating decoys
process could be faster if the target peptides are isolated based on their masses. There is no way
to find the same sequence of any shuffle (decoy) of such peptide in different hash buckets. All
similar masses are hashed into the same bucket.

Clearly, because of the “N”, “C”, and “NC” formats discussed above, isolating, or
localizing the processes based on amino acids letters (or we can say alphabetically) is not
sufficient. As a counter example, if one thread has all the peptides starting with “A” amino acid
(Alanine), then to check the validity of a candidate decoy, an inter-thread communication must
be established since the local set of peptides is not enough. The same decoy or target could be
generated in a different thread, or threads. Moreover, given only 20 amino acids that constitute
the peptides and proteins sequences, this solution could not be a good choice in terms of
scalability, since we have a limited number of amino acid letters (only 20). For partitioning into
more groups, we may use next letters in such sequence which will increase the complexity. One
could suggest that, say, ASCII codes of amino acids letters could be used in hashing, but this option, unfortunately, could also be invalid since the amino acid letters can occur frequently in several places within even the same peptide, which simply means incorrectly generating the same signature/hash value of different peptides.

Figure 6 shows the parallelizing steps of the first phase. To keep the code readable and not to add more complexity to the data structures already used in the serial version, we divide the serial phase into two sub-phases. The process of the first sub-phase reads proteins sequences and then cleaves them, in parallel, into their peptides which are then stored in ordinary/unsorted containers. In the second sub-phase, the threads hash the peptides masses and fill the unique/sorted containers in parallel.

As depicted in Figure 6, the solution for this phase could not gain that much speedup. However, this design will help in speeding up the next phase of decoy generation and validation, which is considered one of the main sources of complexity in the serial version besides the I/O operations. The main idea is on how to localize the generation and the validation processes while keeping the serial model intact. Sorting, removing duplicates, and filtering will be performed locally in each thread. In the second phase, each thread generates and validates decoys locally against local containers. After that, in one merging step, decoys become entries in the general shared heap, which contains the real targets as well.

Three different generating decoys techniques are applied in Crux-Tide, 1: reversing the protein sequences, 2: cleaving a valid decoy for each target peptide, if possible, and 3: to generate decoys for each peptide without validation. The first one could be seen as a simple technique compared to others. However, it is also a complex process that may require some HPC solutions especially when the input protein database is large. The last one still follow peptides
shuffling or simple reversing techniques, but since there is no need to keep low overlaps between targets and decoys, the complexity could be slightly better than the second approach. However, following this approach could cause some controversy since the search engine may report decoys where those are originally real targets (i.e., false positives). In that case, the evaluation will be biased. Therefore, validation process is still an important step and the default technique in Crux-Tide for preparing Target-decoy databases. In parallel and against smaller containers of isolated targets, our multithreading approach breaks the dependency and thus localizes the process.

![Figure 2.6: Parallelizing phase 1.](image)

Obviously, eliminating the communication among threads (albeit using a cleverly designed hashing technique) and merging results at the end is applied in HPC in many applications. Since by clever hashing technique, this approach will make a solution more efficient in our problem domain, then there is no reason to avoid this simple approach. Our parallel approach could have a very good impact on distributed HPC environments. Instead of searching or retrieving candidates from the whole database, Target-Decoys databases with hash-based solution could mitigate and speed up the retrieving/searching phase since small segments are queried. Following the same hashing technique for the observed spectra masses, this
partitioning could mitigate the communications in searching phase if a future distributed searching Crux-Tide is utilized. However, in this solution, we have to merge the peptides into one database to preserve the dependency of searching phase on a single database. Figure 7 above shows the parallel processes of generating decoy peptides and decoy proteins (i.e., decoy FASTA).

2.5 Code and Memory Optimization

Besides the hashing technique, we improve the memory consumption, readability and the understandability of the indexer code. Unfortunately, the recent version(s) of Crux-Tide does not show any improvement in terms of code complexity compared to the early versions. Most of the time, it is not easy to catch the wisdom behind using some variables and data structures in a particular way. For example, the indexer maintains a vector of peptides generated inside the “fastaToPb” function, the function we parallelize, and then deletes the content after few lines of the calling statement.
We noticed that there are many intermediate variables and complex data structures that could be replaced in a better way to improve not only the readability, but also the memory consumption. The motivation of doing this code investigation is the unreasonable amount of memory which the operating system reserves to the Crux-Tide indexer. For example, to index 1 GB of FASTA file, the maximum reserved memory for the sequential indexer process exceeds 23 GB, and exceeds 47 GB for 2 GB of FASTA. This simply indicates that this indexer should be improved in order to be viable in the era of large protein databases.

In order to improve the code readability and to eliminate some intermediate variables and data containers which add some degree of complexity and memory consumption, we redesigned part of the code using some known object oriented concepts and by maintaining pointers to the locations of peptides for each protein. Figure 8 below shows the graphical representation of our improvement.

![Figure 2.8: The new structures and the matrix of hashed peptides.](image)

All protein sequences and their attributes are stored in one vector of protein objects. Each protein object has the addresses of its peptides in the matrix of hashed peptides. In this way, there is no need to maintain another data structure to link proteins with their peptides. The matrix
represents the heart of this solution. The columns represent the threads and the rows are the peptides index or sequence number. Each peptide is hashed into a thread bucket based on its mass value and stored as an object along with its properties. In decoy generation phase, each decoy will be stored along with its real version in the matrix. This step simplifies the decoy proteins generation; instead of looking up target/decoy maps for each target, the process only fetches the decoy sequence directly for each peptide. This design also eases the parallelization of decoy FASTA generation. The above design replaces many intermediate variables and simplifies the code complexity where the interaction among the data structures becomes clearer and easier. This design has a significant impact on the memory consumption (see section 6.1).

2.6 Experiments Environments and Data

Both serial and parallel versions have been executed on the same virtual machine on Google cloud; 24 cores Xeon 2.5 GHz and 150 GB RAM. This machine runs Ubuntu 64 bit 16.04 LTS. The proposed parallel solution for Crux-Tide indexer and decoy generator, applied on Crux-Tide version 3.1, is compared to the original serial solution of the same version using different data sizes. We used protein sequences from “UniRef50” dataset downloaded from http://www.uniprot.org/downloads [22]. The original FASTA file was divided into segments of different sizes, so that we can study and show the performance for both parallel and serial solutions while the FASTA file becomes larger and larger. Table I shows the files’ sizes and the corresponding number of proteins’ sequences.
Table 2-I: Datasets used in the experiments.

<table>
<thead>
<tr>
<th>FASTA file size (GB)</th>
<th>Number of Proteins Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>$2.34 \times 10^4$</td>
</tr>
<tr>
<td>0.25</td>
<td>$5.71 \times 10^4$</td>
</tr>
<tr>
<td>0.50</td>
<td>$1.09 \times 10^5$</td>
</tr>
<tr>
<td>0.75</td>
<td>$1.62 \times 10^5$</td>
</tr>
<tr>
<td>1.00</td>
<td>$2.19 \times 10^5$</td>
</tr>
<tr>
<td>1.25</td>
<td>$2.73 \times 10^5$</td>
</tr>
<tr>
<td>1.50</td>
<td>$3.25 \times 10^5$</td>
</tr>
<tr>
<td>1.75</td>
<td>$3.81 \times 10^5$</td>
</tr>
<tr>
<td>2.00</td>
<td>$4.40 \times 10^5$</td>
</tr>
<tr>
<td>2.50</td>
<td>$5.78 \times 10^5$</td>
</tr>
<tr>
<td>3.00</td>
<td>$7.29 \times 10^5$</td>
</tr>
<tr>
<td>3.50</td>
<td>$8.75 \times 10^5$</td>
</tr>
<tr>
<td>4.00</td>
<td>$1.02 \times 10^6$</td>
</tr>
</tbody>
</table>

2.7 Results and Discussion

This section is divided into two subsections in order to evaluate the two main contributions of this paper; the memory consumption and the parallel version of the target decoy indexer in Crux-Tide search engine. The serial code of Crux-Tide is the base (see Table II) when we compare our results. This is because, to the best of our knowledge, there is no parallel solution to indexing Target/Decoy databases. We believe that researchers can heavily contribute to this field by improving the searching and the post-processing phase of protein search engines. Previously it has been largely ignored, may be due to the infrequent usage of indexing Target/Decoy databases. However, in the era of large protein databases and since these databases are continuously growing, we believe that efficient indexers will add a significant value to the current search engines.
TABLE 2-II: Serial execution times and generated peptides.

<table>
<thead>
<tr>
<th>FASTA file size (GB)</th>
<th>Cleaved Targets</th>
<th>Generated Decoys</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>4.48×10^16</td>
<td>4.07×10^16</td>
<td>87.70</td>
</tr>
<tr>
<td>0.25</td>
<td>1.06×10^17</td>
<td>9.45×10^16</td>
<td>206.15</td>
</tr>
<tr>
<td>0.50</td>
<td>2.16×10^17</td>
<td>1.85×10^17</td>
<td>497.30</td>
</tr>
<tr>
<td>0.75</td>
<td>3.28×10^17</td>
<td>2.73×10^17</td>
<td>725.29</td>
</tr>
<tr>
<td>1.00</td>
<td>4.47×10^17</td>
<td>3.52×10^17</td>
<td>1,136.05</td>
</tr>
<tr>
<td>1.25</td>
<td>5.65×10^17</td>
<td>4.52×10^17</td>
<td>1,272.31</td>
</tr>
<tr>
<td>1.50</td>
<td>6.86×10^17</td>
<td>5.40×10^17</td>
<td>1,617.80</td>
</tr>
<tr>
<td>1.75</td>
<td>8.03×10^17</td>
<td>6.24×10^17</td>
<td>1,807.31</td>
</tr>
<tr>
<td>2.00</td>
<td>9.20×10^17</td>
<td>7.04×10^17</td>
<td>2,217.65</td>
</tr>
<tr>
<td>2.50</td>
<td>1.15×10^18</td>
<td>8.64×10^17</td>
<td>2,895.14</td>
</tr>
<tr>
<td>3.00</td>
<td>1.35×10^18</td>
<td>1.03×10^18</td>
<td>3,455.35</td>
</tr>
<tr>
<td>3.50</td>
<td>1.57×10^18</td>
<td>1.19×10^18</td>
<td>4,183.25</td>
</tr>
<tr>
<td>4.00</td>
<td>1.78×10^18</td>
<td>1.35×10^18</td>
<td>4,827.82</td>
</tr>
</tbody>
</table>

2.7.1 Code and Memory Optimization Evaluation

Code optimization could be beneficial for both parallel and sequential solutions. To fairly evaluate the new data structures and intermediate variables reduction, we first compare the two serial versions, namely the original sequential version performance against our improved version using only one thread. Figure 9 shows the execution times of indexing five different sizes of files from Table I. The single-thread parallel version shows better performance than the serial one in all the five cases. This is more apparent when the FASTA files (i.e., protein files) get larger in size. This result shows the efficiency of the data structure we use to manage the proteins and the peptides locations and information instead of intermediate variables and data structures. Besides the serial code complexity, it is not easy to add more improvements without changing the searching phase since the searching phase of Crux-Tide depends on a particular format and the files generated by the indexer. If one needs to keep the searching phase intact, in our opinion, further enhancements to the current indexer serial code will be limited and some degree of complexity will still be inherited in any parallel solution.
Our code optimization also reduces the memory consumption. The memory footprint is a serious limitation in Crux-Tide indexer. For example, a serial Crux-Tide indexer consumes more than 23 GB of RAM in indexing 1GB of protein sequences (FASTA files). This simply indicates that indexing large files of proteins will be beyond the capacity of many machines available in biologist labs. As can be seen in Figure 10, our solution can index a protein sequences’ file using roughly only two-third of the memory space that is used by the serial code. This significant reduction in memory consumption has a direct impact on the performance of the indexer. Moreover, running the parallel version which reserves less amount of RAM keeps plentiful memory space for other processes. Hence, the machine could host other processes without performance degradation because of memory capacity and swap spaces. For example, 46 GB of RAM are consumed by the serial indexer while indexing 2 GB of FASTA file, whereas around only 30 GB of RAM is consumed by the parallel version to index the same file.
2.7.2 Parallel Solution Evaluation

Figure 11 plots the execution times of serial (original) and parallel versions while increasing the FASTA file size and using different numbers of threads. While the proposed shared-memory OpenMP parallel solution exhibits a reasonable speedup, it is clear that this parallel solution has limited scalability. One source of this limitation could be inherited from the serial logic of the indexer and the I/O operations — all peptides should be generated and organized in their data structures before generation and validation of decoys. Despite limited scalability, our proposed algorithm can still efficiently index large FASTA files in around one third and sometimes even one fourth of the original serial time.

As can be seen from Figure 11, the proposed parallel solution shows the benefit when using multiple threads with large FASTA files. Roughly speaking, a significant difference between serial and parallel times is noticeable when processing 1.0 GB or larger files. Using 4 threads, 1 GB of FASTA is indexed by almost 33% of the serial time. This observation simply indicates that indexing less than 1.0 GB of FASTA does not justify using many threads; 4 could be reasonable. FASTA files of a size less than 0.5 GB might as well be indexed serially.
especially if the hardware is shared and not dedicated for this search engine. For larger than 1.0 GB files, we noticed that more threads are beneficial; 8-16 threads are needed for a better execution time. Despite the scalability issue, this means that few threads, thus less computation resources, can achieve the mission. It could be an efficient solution if the machine should also process other duties besides this tool. Figure 12 shows the effect of varying threads in our solution more closely; in essence larger FASTA files should be fed to the indexer to extract more benefit. Both the plots (left and right) in Figure 12 show that the gap between the curves gets larger when the FATSA file size increases. Based on this, we could extrapolate that, say a FASTA file of size 20GB, will be indexed much faster using 16 threads instead of 8, or using 8 instead of 4, and so on.

![Figure 2.11: Serial and parallel execution times.](image-url)
Figure 2.12: (Left) Comparison between the execution times of parallel version using 4 threads and 8 threads (Right) between 8 threads and 16 threads.

The speedups in Figure 13 clearly support the above discussion. Using only 4 threads, the parallel solution can index the same FASTA file in less than half of the serial time. In most of the cases, 8 threads can do the same job in less than one third of the serial time, and in around one fourth of the serial time using 16 or 24 threads. Note that given the largest file in this experiment (i.e., 4 GB), the parallel solution does not need to use 24 threads. However, the parallel version with 24 threads could be beneficial when indexing much larger FASTA files.

Figure 2.13: Speedups. (Speedup = Serial Time / Parallel Time).
Given few yet enough number of processing cores and the improvement in memory consumption, our proposed solution is more efficient for desktop machines which are readily available on a scientists’ workstation, and for medium sizes of databases (i.e., few Gigabytes). It can be used in laboratories for biological experiments as a cheaper alternative to computing clusters (e.g., Amazon and Google). Multicore machines are available in abundance nowadays and reasonably cheap. It could also save the cost and resources if one decides to run over the cloud since there is no need to request very high specs VMs from cloud providers.

In order to validate the accuracy of the results obtained from our solution, we compare the number of targets and decoys peptides generated by the parallel version against the serial one. The number of cleaved targets from input protein sequences in both versions is same. At times, the parallel version does output relatively few extra valid decoys. To clarify, there is a percentage of peptides of low complexity, for example “ASSSQR”, where random shuffling is not able to generate a valid decoy within the pre-defined threshold value of tries. We noticed that increasing this parameter of tries from, currently 6 in the serial version, to 16 unifies the number of decoys in both the serial and the parallel versions. However, in order to reduce the number of tries, we add consecutive amino acids swapping to generate the decoys. In this case, the above example (i.e. “ASSSQR”) will be “ASSQSR” by the first try in generating decoys. In order to keep the randomness, we randomly pick few amino acids and shuffle them after swapping. These are the main reasons behind the extra valid decoys.
2.8 Conclusion

The basic idea of our design is to localize the process of decoys’ generation and validation. Directly spreading the proteins’ sequences over threads will introduce inter-threads communications. This is a result of the dependency in the validation phase on the other threads’ contents. Because of that, the straightforward solution does not work and even serial execution could give much better results. A clever hashing of peptide’s masses is utilized where each group of peptides have all targets and decoys needed in decoys’ generation and validation. Thus, each thread independently works and runs using also small sizes of containers which enhances the performance of the parallel solution. The parallel version can index the same FASTA file by one third of the serial time using only 4 threads and almost one fourth of serial time using 8 threads. Besides the parallelization, we redesigned part of the serial code so that the memory consumption becomes more efficient. The parallel version can index the same files using around two-third of the memory space that the serial version consumes. Further investigations and experiments are needed to improve the scalability. The improved parallel version is publicly available on https://github.com/wmuwiselab/Protien_HPC.
References


[17] Li Y., Chi H., Xia L., Chu X., Accelerating the scoring module of mass spectrometry-based peptide identification using GPUs. BMC Bioinformatics, 15 (2014) 121


CHAPTER 3

DEEP VS. SHALLOW LEARNING-BASED FILTERS OF MS/MS SPECTRA IN SUPPORT OF PROTEIN SEARCH ENGINES


3.1 Summary

Despite the linear relation between the number of observed spectra and the searching time, the current protein search engines, even the parallel versions, could take several hours to search a large amount of MS/MS spectra, which can be generated in a short time. After a laborious searching process, some (and at times, majority) of the observed spectra are labeled as non-identifiable. We evaluate the role of machine learning in building an efficient MS/MS filter to remove non-identifiable spectra. We compare and evaluate the deep learning algorithm using 9 shallow learning algorithms with different configurations. Using 11 different datasets of more than a million spectra generated from two different search engines, different instruments, different sizes and from different species, deep learning models exhibit their power to filter MSMS spectra efficiently. We also show that our simple feature list is significant where other shallow learning algorithms showed encouraging results in filtering the MS/MS spectra. Our deep learning model can exclude around 45%- 50% of the non-identifiable spectra while losing, on average, only 9%-12% of the identifiable ones. As for shallow learning, algorithms of: Random Forest and Support Vector Machine showed encouraging results, eliminating, on
average, 60%-80% of the non-identifiable spectra while losing a higher percentage around 25%-35% of the identifiable spectra. Deep learning algorithm may be especially more useful in instances where the protein(s) of interest are in lower cellular or tissue concentration, while the other algorithms may be more useful for concentrated or more highly expressed proteins.

3.2 Introduction

Protein identification by database searching of MS/MS spectra is one of the most popular techniques currently used in proteomics. Briefly, in database searching, each experimental spectrum is compared against a set of theoretical candidate spectra retrieved from known peptide databases. Those retrieved theoretical spectra are generated from peptides that have precursor values equal to the observed spectrum precursor peptide mass within a user defined tolerance window (w). For example, given w=3 and observed spectrum of mass 1003.00 Da, all theoretical peptides of masses in the range [1000.00, 1006.00] will be retrieved for the matching process.

The computation load and complexity in this process depend on several influential factors, including but not limited to: the database size, the tolerance window, the precursor value of the observed spectrum, and the number of observed spectra. In terms of database’s size, the typical size of protein databases is rapidly increasing [1]. The larger the database, the more candidates will be possibly retrieved for each observed spectrum. Based on the precursor value of the observed spectrum, the matching could be retrieved from few of the candidates up to millions of them. The advancing technology in mass spectrometry devices adds new challenges to the process of protein identification. Newer devices are able to generate Gigabytes of spectra in a very short time, which leads to more extensive computing in addition to more precise and accurate peptide identification. For example, in one day, Quadrupole-Orbitrap instrument
generates around 14 Gigabytes of raw spectra [2]. One Gigabyte of raw spectra is generated per hour by the earlier generation Q-T and LC MS/MS instruments [3].

Peptide identification from these extensive datasets is a typical Big data problem that handicaps the current protein search engines; these engines struggle to handle even small to medium sized observed spectra and databases on a typical workstation. In addition to the size challenge, the current search engines have different abilities in identifying each input spectrum. In other words, coverage is different from one search engine to another for the same input of both spectra and database. Not all input spectra or peaks within observed spectra include important information about the proteins in the original biological sample. A major goal of our research and others is to exclude “possibly irrelevant” spectra in order to not only decrease searching time, but also improve the accuracy of identification.

In the era of Big MS/MS data and for the above reasons, any reduction technique that is able to efficiently remove “possibly irrelevant” spectra will impact the overall searching performance and result quality. Filtering the input spectra using machine learning is not a new topic. However, there is a need for more experiments using different machine learning techniques in order to improve their performance and fairly judge results in the context of MS/MS filtering. Particularly, it is important for the proteomics research community and industry to discuss results using different machine learning algorithms with varying size of MS/MS datasets, different instruments, and from different species, in addition to varying other important parameters.

In this paper, for the first time to best of our knowledge, we introduce and evaluate the deep learning algorithm in building model-based MS/MS filters. We also compare and evaluate different supervised machine learning algorithms that are already published in the literature for
building MS/MS filters. We use 10 different datasets generated from two semantically different search engines (Comet and pFind). Furthermore, these spectra are observed from samples of six different species. Evaluation of spectra from these varying samples makes our comparisons fair and generalizable.

3.3 Background

Advancements in mass spectrometry, coupled with limitations in computing power, create a need in developing more efficient search engines. The number of generated observed spectra can easily reach billions. Protein search engines could take several hours to search this burdensome quantity of spectra. Pre-filtering is the exclusion of bad, noisy or irrelevant spectra, or those that possibly do not have any information about the proteins in the biological sample, from the searching space prior to peptide identification. This study focuses on only the experiments that utilize machine learning algorithms in building filters.

In [4], QDA (Quadratic Discriminant Analysis) and five features identified by biochemistry experts were used to build spectra filter. These results showed good correlation with the results of MASCOT search engine. Another QDA was also built in [5], but with 187 element of spectra peaks, and compared to the SVM (Support Vector Machine) model developed in [6]. SVM results showed that 67%-70% of the bad spectra can be eliminated with this approach, with around 10% of identifiable spectra lost. In [7], Random Forest was utilized in building filters using features calculated from the spectra peaks and others directly from biochemistry experts. This model lost up to 10% of relevant spectra while filtering out 43% - 75% of the bad spectra. On 78 features of different selected biochemistry properties and peaks which could show a strong statistical evidence about its usefulness, work in [8] evaluated
Bayesian classifier and decision tree algorithms. The developed filters were tested on three datasets and removed 38%-79% of the noise spectra, also losing 10% of the identifiable spectra. In [9], each spectrum’s intensities are normalized in order to mitigate the effect of high peaks on the lower ones. SVM was evaluated on this data, with 10% of relevant spectra removed and 75% of bad spectra eliminated. In [10], LDA (Linear Discriminat Analysis) was applied to datasets after searching the mass spectra and filtering the results by PeptideProphet [11]. Each spectrum was represented by more than 40 produced features from the multistep preprocessing phase of observed spectra.

In the context of pre-filtering the spectra in protein search engines and the quality of classification (i.e., filtering into good or bad spectra), results mainly depend on the dataset being used and the instruments that generate the observed spectra [8, 12]. Moreover, each study in the literature suggests a different set of features to train the machine learning models. The sets include vectors of spectra peaks, sum of intensities, standard deviation of the intensities, the most intense peak, the number of peaks, etc. However, since the filtering method would be an integral part of the search engines that could reduce the searching space, the models and the preprocessing to prepare the spectra prior to the testing process should be as simple as possible. Otherwise, the filtering process may not optimize reduction of computation load.

In this paper, we add a few features extracted from the peak list of each spectrum in addition to those features already evaluated in the literature such as the number of peaks, precursor value, the most intense peak, etc. We tested more than 10 algorithms on different datasets searched using two different search engines (Comet and pFind). We demonstrate optimal values that the deep learning algorithm can add in building MS/MS filters compared to those algorithms already evaluated in literature.
3.4 Methods and Materials

Before diving into the details, in this section we clarify the difference between the datasets in section (A) and the datasets in section (D). The dataset in section (A) consists of MS/MS spectra collected in RAW format from different shotgun proteomics’ projects published on PRIDE [13]. We search them using the search engines (Comet and pFind) and the searching results are used to generate the learning datasets in section (D) to build the machine learning models.

3.4.1 Testing and Evaluation Datasets

We used seven different public datasets that were downloaded from the PRIDE repository. Table I shows the properties of the datasets and the download information.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accession #</th>
<th>Instrument</th>
<th>Tissue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human01</td>
<td>PXD003651</td>
<td>LTQ Orbitrap</td>
<td>prostate epithelium</td>
</tr>
<tr>
<td>Human05</td>
<td>PXD005390</td>
<td>LTQ</td>
<td>epithelial cell</td>
</tr>
<tr>
<td>Human06</td>
<td>PXD004193</td>
<td>LTQ Orbitrap XL</td>
<td>NA</td>
</tr>
<tr>
<td>Mouse</td>
<td>PXD005330</td>
<td>LTQ Orbitrap</td>
<td>NA</td>
</tr>
<tr>
<td>Rat</td>
<td>PXD004150</td>
<td>Q Exactive</td>
<td>brain</td>
</tr>
<tr>
<td>Soybean</td>
<td>PXD001943</td>
<td>LTQ Orbitrap Velos</td>
<td>embryonic axis</td>
</tr>
<tr>
<td>Human-HEK293</td>
<td>PXD001468</td>
<td>Q Exactive</td>
<td>brain, kidney</td>
</tr>
</tbody>
</table>

The code beside dataset names is to distinguish them through the experiments and discussion. As depicted in Table I, seven different datasets are used in our experiments. It is
important to conduct searches on a variety of samples and instruments so we reduce dependency of the results on these factors and evaluate the robustness of our filtering method.

3.4.2 Supervised Machine Learning Algorithms

We evaluated 10 different supervised machine learning algorithms, namely: (1) SVM (Support Vector Machine) using two different Kernel functions (Linear, Radial). (2) Naïve Bayes. (3) Random Forest using three different random configurations of the number of trees (300, 500, 1000). (4) KNN (K-nearest neighbors) using three different random values for k (3, 13, 33). (5) Logistic regression. (6) Artificial (Traditional) Neural Networks using four random values of unit size in the hidden layer (5, 25, 50, 75). (7) LDA (Linear Discriminant Analysis). (8) QDA (Quadratic Discriminant Analysis). (9) Decision Trees. (10) Deep Learning. We developed all filtering programs using R. Deep learning has shown substantial success in different domains and it has become a buzzword nowadays, with several researchers evaluating its capabilities in their respective fields of research. One major issue with deep learning is its dependency on the configuration of its parameters. Fortunately, we recently published a configuration method using Particle Swarm Optimization (PSO) to maximize the overall accuracy of deep learning algorithm [14]. Manual grid search was utilized to optimize the parameters of the other learning algorithms.

3.4.3 Protein Search Engines

We used two semantically different search engines to generate the datasets for the machine learning algorithms. We searched the datasets in Table I once using Comet and its Percolator [15] and once using pFind [16]. The searching was against two different databases,
namely Uniprot_sprot and ipi.Human.v3.87. The objective of using two search engines is to make our results generalizable and to study the performance of each machine learning algorithm in different conditions. We also suggest that this approach is followed in this kind of experiment unless we build specific products for specific conditions.

3.4.4 Datasets for Machine Learning

After searching, each spectrum is represented by 10 features that are extracted from its m/z and intensity information. Each spectrum is also labeled as “identified” or “not identified” based on the results of the search engines. Table II shows the features that we used in this study. Many of them are already used in the literature; for example, the mean and standard deviation.

<table>
<thead>
<tr>
<th>#</th>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Precursor</td>
<td>Precursor m/z value of each spectrum.</td>
</tr>
<tr>
<td>2</td>
<td>Mean(intensities)</td>
<td>The average of intensities.</td>
</tr>
<tr>
<td>3</td>
<td>SD(intensities)</td>
<td>The standard deviation of intensities.</td>
</tr>
<tr>
<td>4</td>
<td>Int_above_mean</td>
<td>Number of peaks greater than Feature 2.</td>
</tr>
<tr>
<td>5</td>
<td>Above mean ratio</td>
<td>Feature 4 / total number of peaks (Feature 7).</td>
</tr>
<tr>
<td>6</td>
<td>Entropy(intensities)</td>
<td>Entropy is the measure of randomness of all peaks for each spectrum.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$= -\sum p_i \log p_i$, $p_i$ is the intensity of peak i.</td>
</tr>
<tr>
<td>7</td>
<td>Number of peaks</td>
<td>Number of peaks in each spectrum.</td>
</tr>
<tr>
<td>8</td>
<td>Range of m/z values</td>
<td>Max(m/z) – Min(m/z)</td>
</tr>
<tr>
<td>9</td>
<td>Mz_of_max</td>
<td>The m/z value of the highest peak.</td>
</tr>
<tr>
<td>10</td>
<td>The m/z of the most left peak.</td>
<td>The m/z value corresponds to the most left peak (lowest m/z) after applying top 200 intense peaks.</td>
</tr>
</tbody>
</table>

As for searching parameters, we have used the same parameters for both Comet and pFind search engines. For each search engine, not all observed spectra are identified. This makes our datasets imbalanced in terms of our class label (i.e. identified or not). This is related to the coverage and other issues which are out of the scope to be discussed in this study. Sometimes more than 70% of the peaks are identified, while on some search engines less than 30% of them
are identified. To make the comparisons fair for all algorithms, we balance the dataset by using an “undersampling” technique. This makes the random decision as identified spectrum or not to have the probability of 50% in our analysis. This process of building the datasets for machine learning models resulted in 10 different datasets. The models and the used features are also evaluated using another very large and more representative dataset. HEK293, it consists of more than a million spectra generated by Q Exactive spectrometer. For more details and information about this dataset, one can refer to PRIDE website using the accession number in Table II, above. Table III shows the dataset and their sizes (i.e. number of spectra). The size of HEK293 is based on Comet search engine v 3.1.

<table>
<thead>
<tr>
<th>#</th>
<th>Dataset Name</th>
<th>Search Engine</th>
<th>Dataset Size (# of spectra)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Human01</td>
<td>pFind</td>
<td>5,656</td>
</tr>
<tr>
<td>2</td>
<td>Human01</td>
<td>Comet</td>
<td>12,544</td>
</tr>
<tr>
<td>3</td>
<td>Human06</td>
<td>pFind</td>
<td>2,738</td>
</tr>
<tr>
<td>4</td>
<td>Human06</td>
<td>Comet</td>
<td>3,284</td>
</tr>
<tr>
<td>5</td>
<td>Human05</td>
<td>Comet</td>
<td>8,690</td>
</tr>
<tr>
<td>6</td>
<td>Mouse</td>
<td>pFind</td>
<td>16,088</td>
</tr>
<tr>
<td>7</td>
<td>Mouse</td>
<td>Comet</td>
<td>5,720</td>
</tr>
<tr>
<td>8</td>
<td>Soybean</td>
<td>pFind</td>
<td>8,076</td>
</tr>
<tr>
<td>9</td>
<td>Soybean</td>
<td>Comet</td>
<td>21,000</td>
</tr>
<tr>
<td>10</td>
<td>Rat</td>
<td>Comet</td>
<td>12,000</td>
</tr>
<tr>
<td>11</td>
<td>Human-HEK293</td>
<td>Comet</td>
<td>1,119,064</td>
</tr>
</tbody>
</table>

3.4.5 Models Evaluation

For the evaluation process of machine learning algorithms, each dataset is randomly shuffled then divided into 75% training and 25% unseen testing datasets. Each algorithm is fed with the same datasets so that we make fair comparisons. For evaluation, we use three metrics
which are used together in evaluating classifiers: the overall accuracy, sensitivity, and specificity. There are many online resources that show the simple calculation for each one of them. Here we define the meaning of each classifier in the context of MS/MS filtering. The overall accuracy shows the percentage of MS/MS spectra that are correctly classified as: identified or not identified. However, this measure may mislead the research community. For example, matching with 50% accuracy may be misleading because of the biased classification towards one class (say 100% True Positive and 0% True Negative). Therefore, we need to show the sensitivity, which shows how many times the classifier labels the real identifiable spectra as identifiable class (i.e., actually identified = yes, predicted = yes). The specificity shows how many times the classifier labels the real Non-identifiable spectra as Non-identifiable class (i.e., actually identified = NO, predicted = NO). For MS/MS filters, sensitivity is defined as the percentage of identifiable spectra retained in the system, and similarly, specificity means the percentage of non-identifiable spectra removed from the system. For example, a filter of sensitivity of 90% and specificity of 44% means it is able to correctly exclude 44% of the non-identifiable spectra while losing 10% of the identifiable ones.

3.4.6 HEK293 Dataset Processing

HEK293 dataset consists of 24 raw peak files of total size around 71 GB. For more efficient computation and analysis compared to the serial solutions on one machine, a cluster of computing nodes is utilized to handle this dataset. Each computing node runs the same pipeline which processes a file at a time for all targets of this chapter and the subsequent chapters. For example, each computing node searches a file using Comet, filters the results using percolator, processes the spectra of the file for machine learning (i.e. extract 10 features), then evaluate the
models of different machine learning algorithms. In the next stage of the pipeline, each file’s spectra will be processed and sampled to generate the reduced files that Comet will search again and compare for the techniques and ideas of chapter 6, and so on.

Our cluster has 12 machines connected via local area network (LAN) and all of them run Linux OS and access a shared storage. This cluster is homogenous; all computing nodes have the same computation power, or in different words, have the same hardware specification.

Processing MS/MS files in distributed environments is discussed in literature in several research works. Different distributed solutions depend mainly on spreading the MS/MS files or set of spectra over the computing nodes. Section 1.5 showed how distributed computing environments could be efficient solutions to tackle the large amount of MS/MS spectra and also the large databases of protein sequences. The distribution of workload across multiple computing nodes, for instance, could maximize the throughput, minimize the probability of failure due to hardware and software limitations and could also have high availability. In developing machine learning solutions for big data, the distributed approach could be the choice to overcome some of the limitations of the traditional algorithms. For example, several researches are currently discussing the distributed deep neural networks to utilize the computation resources and to cope with the massive amount of data currently being generated by real world applications [17-18].

There is no doubt that the size of the training/testing data is an important factor in machine learning applications. Due to our distributed processing of the dataset, the models are built per file. To evaluate the distributed process, the vectors of features extracted from multiple files (randomly picked) are combined together; 2 files, 4 files and 6 files. Compared to the results in figure 13, below, Naïve Bayes and QDA are the algorithms that showed significant drop in
their accuracy, sensitivity and specificity; more than 30% in their sensitivity values. Regarding other algorithms, there are no significant changes in the performance.

3.5 Results and Discussion

This section is organized based on evaluation of the datasets as ordered in Table III above. In figures 1-10, we show the performance of each algorithm used in this study on the first 10 datasets. Regarding the HEK293 dataset, Figure 11 (A –X) show the performance of the used algorithms on the individual dataset files. Figures 12 and 13 are the summary of the values using 95% confidence interval.

3.5.1 Human01 Dataset

Figure 1 depicts the performance of the ML algorithms on Human01 dataset generated from the pFind search engine. Clearly, we notice that Random Forest with different numbers of trees is able to filter out more than 86% of the non-identifiable spectra while it loses around 14% of the identifiable ones.
Decision trees can also remove more than 75% of non-identifiable spectra while losing about 16% of identifiable ones. KNN algorithms perform less accurate than DT; QDA and LDA follow KNN in accuracy. The figure also shows the performance of deep learning with PSO optimization. While our deep learning model can remove around 70% of the non-identifiable spectra, less than 7% of identifiable spectra are removed. In Figure 2, none of the algorithms surpass deep learning. The model removes around 45% while losing 12% of the identifiable spectra.

QDA, in Figure 3, eliminates more than 55% of non-identifiable ones, but also loses around 16% of the identifiable ones. Compared to other shallow algorithms, QDA performs better. The performance is very close to deep learning model in this dataset. Deep learning model eliminates about 10% more of non-identifiable spectra. In Figure 4, in fact, many of them perform well, including SVM, and NN (Neural Networks) of size 5 and NB (Naïve Bayes), KNN when k = 33, QDA and LDA. Using the deep learning model, we remove less non-identifiable spectra compared to other algorithms without losing more than 7% of the identifiable ones.
3.5.2 Human06 Dataset

QDA, in Figure 3, eliminates more than 55% of non-identifiable ones, but also loses around 16% of the identifiable ones. Compared to other shallow algorithms, QDA performs better. The performance is very close to deep learning model in this dataset. Deep learning model eliminates about 10% more of non-identifiable spectra. In Figure 4, in fact, many of them perform well, including SVM, and NN (Neural Networks) of size 5 and NB (Naïve Bayes), KNN when \( k = 33 \), QDA and LDA. Using the deep learning model, we remove less non-identifiable spectra compared to other algorithms without losing more than 7% of the identifiable ones.
3.5.3 Human05 Dataset

Random Forest, SVM, KNN (K=33), and DT (decision Tree) perform better than the rest of evaluated algorithms. The deep learning model is still biased towards the identifiable spectra and shows higher sensitivity than others; around 94%, as demonstrated in Figure 5. However, the former algorithms eliminate more non-identifiable spectra. In this case the selection will depend on the application; if the search engine is robust enough to find the proteins even without 15%-17% of the identifiable spectra, then deep learning may not be the choice for this dataset. Otherwise, with the deep learning algorithm we reject only 46% of the non-identifiable spectra losing only 6% of the identifiable ones. The deep learning algorithm may be especially more useful in instances where the protein(s) of interest are in lower cellular or tissue concentration, while the other algorithms may be more useful for concentrated or more highly expressed proteins.
3.5.4 Mouse Dataset

For the dataset generated by pFind (Figure 6) deep learning shows better results than others. It is followed in performance by QDA and Naïve Bayes. Despite the balance on the other models between sensitivity and specificity, the lost ratio is higher than the former three models. The case is different with the Comet dataset, although the deep learning model still performs better; Random Forest, Neural Network, Decision Trees and Support Vector Machine (SVM) follow in accuracy after the deep learning model.
3.5.5 Soybean Dataset

Figure 8 shows that deep learning model perform much better than others. While eliminating around 50% of the non-identifiable spectra, 12% of the identifiable are lost. Figure 9 shows that SVM, Random Forest, Neural Network, and LDA lose around 18% of the identifiable spectra, but they can eliminate around 70% to 75% of the non-identifiable ones.

The deep learning model is still a conservative one towards the sensitivity and losing up to 8% of the identifiable spectra, while excluding around 53% of the non-identifiable ones.
3.5.6 Rat Dataset

In Figure 10, the deep learning model still reports the highest sensitivity, but does not eliminate more than 32% of the non-identifiable spectra while losing 9% of the identifiable ones. QDA has similar performance, followed by Naïve Bayes and Support Vector Machine (SVM).

3.5.7 HEK293 Dataset

Figure 11 (A-X) shows performance of the algorithms used in this study on the spectra of HEK293 files.
(A) b1906_293T_proteinID_01A_QE3_122212.raw

(B) b1922_293T_proteinID_02A_QE3_122212.raw

(C) b1923_293T_proteinID_03A_QE3_122212.raw
(G) b1927_293T_proteinID_07A_QE3_122212.raw

(H) b1928_293T_proteinID_08A_QE3_122212.raw

(I) b1929_293T_proteinID_09A_QE3_122212.raw
(J) b1930_293T_proteinID_10A_QE3_122212.raw

(K) b1931_293T_proteinID_11A_QE3_122212.raw

(L) b1932_293T_proteinID_12A_QE3_122212.raw
(M) b1937_293T_proteinID_01B_QE3_122212.raw

(N) b1938_293T_proteinID_02B_QE3_122212.raw

(O) b1939_293T_proteinID_03B_QE3_122212.raw
(P) b1940_293T_proteinID_04B_QE3_122212.raw

(Q) b1941_293T_proteinID_05B_QE3_122212.raw

(R) b1942_293T_proteinID_06B_QE3_122212.raw
Figure 3.11: The performance of machine learning algorithms using HEK293-Comet dataset.
In general, it is no surprise that with different datasets, the performance of machine learning algorithms varied. The above experiments reveal that Random Forest, SVM, Neural Network, QDA, and KNN usually report encouraging results. However, logistic regression always fails to maintain comparable accuracy to others that are tested in this experiment. The results also showed experimental evidence that deep learning performs the filtering task efficiently. Deep learning eliminates around 45%-50% of the non-identifiable spectra while losing around 9%-12% of identifiable spectra. This is considered a good improvement to the current search engines, especially in terms of sensitivity. Of course, sometimes other algorithms such as Random Forest and QDA are similar to deep learning in terms of efficiency, but not in all datasets. Figures 12 and 13 show the overall performance of the algorithms of our study on different datasets. The figures depict the overall average accuracy, average sensitivity, and the average specificity of the models on the 11 datasets. We also show the results with the error rate using a 95% confidence interval. Figures 12 and 13 summarize the above experiments. Random Forest, SVM, KNN, and Deep learning could efficiently help the search engines as an ensemble of searching space reduction techniques. We could favor deep learning models over the others because of its sensitivity. Reducing around 45%-50%, on average, of the all non-identifiable spectra while, on average, losing only 9%-12% of identifiable spectra should significantly enhance the performance of search engines. Regarding the models of shallow learning algorithms, although reducing around 60%-80% of non-identifiable spectra, on average, could be better; those methods lose, on average, 25%-35% of the identifiable spectra, and sometimes more than 35%. These losses could be unacceptable when identifying low concentration proteins or those difficult to detect. Moreover, the models of deep learning are strong competitors of the other shallow learning algorithms in all datasets as depicted in the above figures.
Two more important points to discuss: There is no need to discuss algorithmic speed of identification in this study since the models built performed all calculations in a few seconds. This could be one great advantage for building light models with a few numbers of features. Secondly, in this study we tried to combine features with the original peak intensity values, but that combination does not significantly improve model accuracy. Sometimes the combination showed 1%-2% improvement which may not be significant enough to warrant a heavier feature model. We favor to build lighter models with few features since 2% increase is not worth increasing the complexity of the algorithm. However, sometimes the accuracy gets worse with a long list of features.

Figure 3.12: Average performance of various machine learning algorithms across the first 10 datasets in our experiment.
3.6 Conclusion

In this study, we introduced the deep learning model in building MS/MS filters as an efficient technique to reduce searching space. This study supports the quality of search engine results and performance. We compared 10 different supervised machine learning algorithms using different configurations. The comparisons are conducted using 9 shallow learning algorithms with different configurations against deep learning models where the Particle Swarm Optimization controls the deep learning parameters, namely the numbers of layers and neurons. We also added some features to those already used in the literature. We used features that can be easily extracted from spectra peaks. Using 11 different and large datasets, one even more than a million spectra, generated from two different search engines, different instruments, different species and different sizes, we showed experimental evidence that deep learning models are powerful in filtering MS/MS spectra. We also showed that our feature list is significant where other shallow learning algorithms showed encouraging results in filtering the MS/MS spectra. Those features are very simple and can be extracted in a short and consistent time. Deep learning provided the best overall results with respect to the highest sensitivity -- very important in identifying low concentrations of proteins or those more difficult to identify. Regarding the prediction time, for all models, it does not exceed few seconds on all unseen testing datasets.
the future, as a lesson of this work, we will focus on deep learning, SVM, Neural Networks, and Random Forest in trying to design more efficient filters where training and testing datasets have millions of observed spectra.
References


CHAPTER 4

OPTIMIZING PROTEIN SEARCH ENGINES USING PARTICLE SWARM OPTIMIZATION

This chapter presents the contribution of my publications: Optimizing Protein Search Engines using Particle Swarm Optimization. Proceedings of the 17th International Conference on Bioinformatics and Bioengineering (IEEE BIBE 2017).

4.1 Summary

The results of protein search engines depend mainly upon a set of parameters that adjust the searching space. One of the most effective parameters is the peptide mass window tolerance (w). Most of the current search engines use a constant user-defined value for this parameter. As an alternative option, Comet search engine designers proposed a statistical technique to estimate the best tolerance window for an input spectra file. However, this technique sometimes fails in picking a value, may set the parameter to a value that results in a loss of many correct matches, and is available only for one type of mass; namely ppm. In this paper, we propose to use particle swarm optimization (PSO) to improve the coverage of search engines by picking the optimal value for this influential parameter to maximize PSMs. Our results show that this biologically-inspired algorithm can be utilized to find peptide mass window tolerance values that facilitate Comet to increase peptide spectra matches, resulting in improved peptide identification. We also show experimental evidence that an open search (i.e., wide tolerance window) does not always...
optimize spectra matching using the current search engines and that narrow tolerance windows improve the coverage of protein search engines.

4.2 Introduction

Mass spectrometry-based proteomics is one of the most popular techniques in protein identification. In terms of number of publications, Google scholar shows that there are 12,300 publications since the year 2000 in which the keywords "mass spectrometry proteomics", appear together. 70% of those papers were published since 2010, and 23% since 2015. This abundance of work indicates the heavy dependency and usage of this process in research fields such as computational biology, medical sciences, etc.

The fundamental task in mass spectrometry-based proteomics is the identification of peptides, fragments of proteins comprised of linked amino acids. There is no single, straightforward step to detect amino acids sequences directly from the biological samples. The protein is identified by matching experimental mass spectrometry spectra with theoretical or database spectra. In general, there are three basic techniques to identify/match the peptides from mass spectra: (1) De novo method, where the amino acids are identified based on direct calculations of the spectrum’s peaks, (2) Spectral library approach in which each input spectrum is searched against previously identified spectra, and the most common one is (3) Searching the input spectra against protein database(s) [1-3]. Figure 1 shows the major tasks and components of a typical observed spectra search engine.

As figure 1 depicts, based on the precursor mass of the input spectrum, the engine retrieves a set of candidate peptides from the peptides’ database. Candidate peptides are those peptides which have precursor values equal to the observed spectrum precursor value ± w, where
w is a user defined window tolerance of peptides’ mass. For example, given w=3, an experimental spectrum of precursor value of 700 will be compared against all theoretical peptides of masses in the range [697, 703].

![Figure 4.1: General SEQUEST-style in searching observed spectra.](image)

Clearly, the performance of scoring function(s), which match and rank the pairs (i.e., observed spectrum and its candidates), depends on several factors. The database size, the precursor value and the window tolerance (w) affect both the load of computations and the number of possible correct peptide-spectrum matches (PSMs). Since the results of scoring functions are the main input of the post-processing phase, in which matches are filtered, those parameters have a direct effect on the final PSMs. In fact, increasing/changing any of these three parameters could result in
a much larger set of candidates for the matching process of an observed spectrum, thereby making the post-processing phase computationally intensive. Therefore, optimization of these parameters will improve a search engine’s capability and directly impact the quality of results.

The peptide tolerance window parameter \((w)\) is usually set as a constant value in most of the currently available search engines. Current search engines depend on a rule of thumb value. For example, in Comet [4, 5] and Crux-Tide [6] the default value is 3 Da. Recently, Crux-Tide and Comet tools are enriched with a clever algorithm that estimates the best value of this parameter \((w)\) based on the input spectra file. The Param-Medic depends on the results of analyzing pairs of input spectra that most likely represent the same peptide. An interesting discussion and more details about this algorithm can be found in [7].

However, despite its fast calculation speed, this method has several limitations: first, it can work only with a specific unit of mass; ppm (Parts per Million). Second, it cannot always reliably estimate the parameters. Third, sometimes, it estimates a value of \((w)\) that is less optimal than the initial value in terms of the final number of matches. This means more missed correct peptides, or PSMs. As a more promising option, we propose Particle Swarm Optimization (PSO) to estimate the best value of \((w)\). The best value, in this context, is the value which maximizes the number of correct matches. Our experiments using 9 different public datasets show encouraging results- the PSO outperforms the rule of thumb method and also returns more precise values of \((w)\) compared to Param-Medic. In addition, our proposed method can easily work with different units of masses in addition to ppm. The experiments also show that the number of matches increased when the values of \(w\) suggested by our PSO algorithm were used.

The rest of the paper is organized as follows. The Particle Swarm Optimization (PSO) algorithm is discussed in the next section. In section III, we discuss our experimental setups
including the datasets, methods and experiment environment. Section IV contains the results and
discussion. We also explore some related works in section V and conclude in section VI.

4.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) has extensive capabilities of global optimization in
addition to easy implementation, scalability, robustness, and fast convergence. It employs simple
mathematical operators and is computationally inexpensive in terms of both memory
requirements and speed [8]. It is an iterative computational optimization method developed by
Kennedy and Eberhart in 1995 [9], and is inspired by social behavior of bird flocking or fish
schooling. In PSO, the set of candidate solutions to the optimization problem is defined as a
swarm of “particles”. Each particle maps the attributes of a solution. Each particle has its position,
velocity and a fitness value that is determined by an optimization function to assist the quality of
the particle. The particle velocity determines their direction and distance of fly. Each particle
updates its velocity and position according to the personal extreme (\(W_i^{\text{best}}\)), which is the optimal
solution that particle finds itself at present, and the global extreme (\(G_{\text{Wbest}}\)) which is the optimal
solution that all particles find at present. Each particle in PSO should regard the current position,
the current velocity, the distance to \(W_i^{\text{best}}\), and the distance to \(G_{\text{Wbest}}\) in order to modify its
position. PSO in our experiment was mathematically modeled as follows:

- **Velocity of Window size**
  \[
  V_{W,i}^{t+1} = a \cdot V_{W,i}^t + c_1 \cdot \text{rand.} \cdot (W_i^{\text{best}} - V_{W,i}^t) + c_2 \cdot \text{rand.} \cdot (G_{\text{Wbest}} - V_{W,i}^t)
  \]

  Where, \(V_W\) is the velocity of the window size, \(W_i^{\text{best}}\) is the particle’s best local value of
  the peptides mass tolerance window size, and \(G_{\text{Wbest}}\) is the best global value of the peptide mass
tolerance window size. \(c1\) and \(c2\) are acceleration coefficients and they present the weights of
statistical acceleration items in approaching to $W_{i}^{\text{best}}$ and $G^{W_{\text{best}}}$ of a particle. $\alpha$ is inertia coefficient, and it controls the flying of particles. *rand* is a uniform random value between 0 and 1. In every step $t$, a position of particle $i$, $W_{i}^{t}$, is updated based on the particle’s velocity $V_{W_{i}}^{t+1}$.

- **Position of Window size**

\[
W_{i}^{t+1} = W_{i}^{t} + V_{W_{i}}^{t+1}
\]  

(2)

![Figure 4.2: The schematic of our PSO model.](image)

Figure 2 shows the schematic of our PSO model in searching for the best value of $w$. In our model we search for the value of $w$ that maximizes number of final matches reported by the percolator semi-supervised machine learning algorithm, improving discrimination between correct and incorrect matches.
Table 4-I: MS/MS spectra and the protein databases.

<table>
<thead>
<tr>
<th>MS/MS Data Set</th>
<th>Accession</th>
<th>Instrument</th>
<th>Organism</th>
<th>Tissue</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human01</td>
<td>PXD003651</td>
<td>LTQ Orbitrap</td>
<td>Homo sapiens (Human)</td>
<td>prostate epithelium</td>
<td>Uniprot_sprot</td>
</tr>
<tr>
<td>Human02</td>
<td>PXD004713</td>
<td>Q Exactive</td>
<td>Homo sapiens (Human)</td>
<td>urine</td>
<td>Uniprot_sprot</td>
</tr>
<tr>
<td>Human03</td>
<td>PXD002394</td>
<td>Q Exactive</td>
<td>Homo sapiens (Human)</td>
<td>cell suspension culture</td>
<td>Uniprot_sprot</td>
</tr>
<tr>
<td>Human04</td>
<td>PXD001424</td>
<td>LTQ Orbitrap Elite</td>
<td>Homo sapiens (Human)</td>
<td>iris</td>
<td>Uniprot_sprot</td>
</tr>
<tr>
<td>Human05</td>
<td>PXD005390</td>
<td>LTQ</td>
<td>Homo sapiens (Human)</td>
<td>epithelial cell</td>
<td>Uniprot_sprot</td>
</tr>
<tr>
<td>Human06</td>
<td>PXD004193</td>
<td>LTQ Orbitrap XL</td>
<td>Homo sapiens (Human)</td>
<td>NA</td>
<td>ipi.Human.v3.87</td>
</tr>
<tr>
<td>Mouse01</td>
<td>PXD005330</td>
<td>LTQ Orbitrap</td>
<td>Mouse</td>
<td>mus musculus</td>
<td>ipi.Mouse. v3.68</td>
</tr>
<tr>
<td>Soybean</td>
<td>PXD001943</td>
<td>LTQ Orbitrap Velos</td>
<td>Soybean</td>
<td>embryonic axis</td>
<td>GLYCINE</td>
</tr>
<tr>
<td>Rat</td>
<td>PXD004150</td>
<td>LTQ Orbitrap Velos</td>
<td>Rat</td>
<td>brain</td>
<td>Uniprot_sprot</td>
</tr>
</tbody>
</table>

4.4 Methods

4.4.1 PSO Model

We use the parameters in Table II to configure the PSO model to search for the best value of window tolerance.

Table 4-II: The utilized parameters in our experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>Def. 5</td>
</tr>
<tr>
<td></td>
<td>ppm 10</td>
</tr>
<tr>
<td>Learning coefficients: c1, c2</td>
<td>uniformly distributed between [0, 2]</td>
</tr>
<tr>
<td>Maximum number of iterations</td>
<td>5</td>
</tr>
<tr>
<td>Window size</td>
<td>Def. within the range [1, 15]</td>
</tr>
<tr>
<td></td>
<td>ppm within the range [1, 50]</td>
</tr>
<tr>
<td>Window size velocity</td>
<td>MinWindowVelocity= -0.05(MaxWindow - MinWindow)</td>
</tr>
<tr>
<td></td>
<td>MaxWindowVelocity= +0.05(MaxWindow - MinWindow)</td>
</tr>
</tbody>
</table>

In our experiments, we compare our PSO model results against both the default value of peptide window tolerance; (i.e., ±3 Da.) and the values optimized by the Param-Medic tool described in [7]. Because the Param-Medic tool works only with one unit of mass, ppm, we run
PSO twice: the first one with the default values of mass unit (Dalton) and mass tolerance \((w=3.0)\) Dalton), and a second time using a different mass unit, ppm, so that we can evaluate our results against the results of \textit{Param-Medic}. We use different ranges of the maximum number of iterations and the window size due to different value ranges between Dalton and ppm.

In both settings, the algorithm starts by running Comet and Percolator using random numbers from a predefined range of window size. After that, the global best variable is updated by the window size which reports highest number of matches. The other swarm will then move toward this value based on the velocity formula. Once the improvement in number of matches is not significant, the PSO terminates the optimization process and report the best value of tolerance window \((w)\). In our experiment, the PSO terminates the process if there are no extra matches reported by the local swarms over the global value. As seen in table II, we use different population size with different Comet parameters. We use 5 swarms with Da mass unit and 10 with ppm mass unit. The difference in used number of swarms is because for the ppm mass unit, the searching space is larger and needs more resources to find more accurate solutions; see the ranges in Table II.

4.4.2 Data Sets

To fairly evaluate the PSO in this context, we selected nine public data sets of different instruments, tissues or species from the PRIDE repository [10]. Table I shows details about the MS/MS data sets and their sources. We use sequence numbers with the same data set name, e.g. human01, human02, etc. We also use multiple protein databases in order to show the robustness of our model. We test different combinations of data sets to show the versatility of our model.
and encourage stakeholders in the proteomics research community to use and build upon this method.

Although our proposed solution is data independent, another evaluation on larger dataset is also conducted using HEK293 dataset. Using HEK293 datasets has two main objectives; to evaluate the time required for larger MS/MS files and to compare the results of PSO with Param-Medic and also with fixed value, 20 ppm, used in Comet search engine evaluation on human dataset. HEK293 consists of more than a million spectra generated from Q Exactive spectrometer. For more details and information, one can refer to PRIDE dataset using the accession number PXD001468. The HEK293 dataset results and evaluation are discussed separately in section 4.5.

4.4.3 Experiments Environment and Comet Parameters

We run Comet using 8 threads on our lab workstation; Intel Xeon(R) CPU E5-1607 v3@ 3.10 GHz, and 64 GB of RAM, which runs 64-bit Ubuntu 16.04 LTS.

For HEK293 dataset, ppm window type parameter fragment_bin_tol of 0.02 is used, this is a recommended parameter for high resolution data by Comet designers.

4.5 Results and Discussion

In the first part of this section, we experimentally show the limitations of the Param-Medic tool in setting the tolerance window parameter and how our PSO overcomes these limitations. In the second part, we compare the performance of Comet using both the tolerance window recommended by our PSO model and the one recommended by Param-Medic to
performance using the default value of 3. After that, we compare our model and Param-Medic in terms of both the performance and the limitations.

As depicted in Table III, our PSO model successfully picks the best w for all datasets, whereas, Param-Medic cannot find the solution for many of these MS/MS datasets and returns the output “fail.” Param-Medic did not find sufficient information from some of the input files as shown. Our PSO model does not only rely on the content of files, but rather runs Comet using a few values of w. As a result, our technique always successfully returns a value with either minor improvement or without improvement if the default value turns out to be the best. As mentioned above, we randomly chose the MS/MS files to cover different instruments and proteomics projects in PRIDE. Unfortunately, The Param-Medic tool was able to set the parameter for only five out of nine MS/MS input files. Certainly, improving this tool to overcome this limitation will be a significant improvement in mass spectra searching.

In terms of matching results, Comet was able to report more PSMs using the PSO model-recommended w values compared to the Param-Medic recommendations. For two datasets, namely, human01 and Rat projects, the improvement is clearly significant. These results indicate that the searching phase was able to find a large set of candidates with correct matches using our PSO model recommendations. The percolator, in turn, was able to find higher quality matches. For all datasets above, Param-Medic tends to recommend wider tolerance windows than our PSO, resulting in a larger searching space. Based on the results (PSMs), this larger search space is inefficient, resulting in many random matches and decreasing the final correct matches.
Table 4-III: Comparison between Comet results when using the recommended \( w \) by our PSO model and the Param-Medic tool

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PSO PSMs</th>
<th>PSO Best ( w ) (ppm)</th>
<th>Param-Medic PSMs</th>
<th>Param-Medic Best ( w ) (ppm)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human01</td>
<td>1735</td>
<td>9.38</td>
<td>1621</td>
<td>35.6</td>
<td>+7%</td>
</tr>
<tr>
<td>Human02</td>
<td>1035</td>
<td>8.41</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Human03</td>
<td>1079</td>
<td>14.85</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Human04</td>
<td>3602</td>
<td>12.2</td>
<td>3505</td>
<td>30.2</td>
<td>+3%</td>
</tr>
<tr>
<td>Human05</td>
<td>644</td>
<td>49.7</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Human06</td>
<td>1351</td>
<td>18.61</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Mouse01</td>
<td>4886</td>
<td>18.1</td>
<td>4837</td>
<td>47.41</td>
<td>+1%</td>
</tr>
<tr>
<td>Soybean</td>
<td>8003</td>
<td>24.81</td>
<td>7903</td>
<td>44.4</td>
<td>+1%</td>
</tr>
<tr>
<td>Rat</td>
<td>850</td>
<td>6.32</td>
<td>555</td>
<td>93.4</td>
<td>+35%</td>
</tr>
</tbody>
</table>

On the other hand, while Param-Medic needs a few seconds to a few minutes with larger files to estimate the tolerance window, our PSO model needs a much longer time. This is because we run Comet and percolator in each iteration of the PSO algorithm. Table IV shows the execution times needed by our PSO to return the best \( w \). The increased execution time of PSO is a disadvantage, but its encouraging results for better \( w \) and increased matches make it an intriguing technique. Furthermore, PSO could be speeded up by utilizing some sampling techniques and parallel computing but that is out of the scope of this paper and remains as a future work. Our main focus here is to evaluate the performance of PSO in boosting the search engines.

Because the Param-Medic tool does not work with all units of peptide mass but only with ppm, we cannot run it using Comet default parameters. The default mass unit in Comet is “amu” (atomic mass unit) or Dalton and the default tolerance window is 3.0 Dalton. Therefore, we use the same Comet results discussed above and compare them once against the results of Comet running on default parameters and once on our PSO recommendation for \( w \) values.
Table 4-IV: PSO execution times.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PSO time in minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human01</td>
<td>127.2</td>
</tr>
<tr>
<td>Human02</td>
<td>121.2</td>
</tr>
<tr>
<td>Human03</td>
<td>56.0</td>
</tr>
<tr>
<td>Human04</td>
<td>114.0</td>
</tr>
<tr>
<td>Human05</td>
<td>54.0</td>
</tr>
<tr>
<td>Human06</td>
<td>68.4</td>
</tr>
<tr>
<td>Mouse01</td>
<td>51.4</td>
</tr>
<tr>
<td>Soybean</td>
<td>148.2</td>
</tr>
<tr>
<td>Rat</td>
<td>65.4</td>
</tr>
</tbody>
</table>

Table V shows the results of Comet using our PSO model recommendations for the tolerance window in Dalton units. In comparison to the Comet results using default parameters, our PSO successfully maintains or increases correct matches identified by Comet. Table V shows that our PSO does not always recommend values which significantly increase the matches, but as in the case of Human05 dataset, a 30% increase in correct matches is significant. PSO tends to recommend narrower window tolerance compared to the default tolerance window and the window recommended by Param-Medic. This is because it starts with normalized random values within predefined range of $w$ values. The swarms iteratively approach the best solution. In most of our experiments, Comet reported more matches using low values of $w$. However, this significant observation needs confirmation using more experiments testing a large set of MS/MS files. In contrast, in terms of speed and the number of final correct matches, Comet performs better on default values compared to its performance using the Param-Medic recommendations. As in the case of “Rat” dataset in Table V, the Param-Medic misleads the Comet engine, with a 21% decrease in correct matches. The Param-Medic tool mostly decreases correct matches by Comet in other trials, as shown in Table V.

Comet runs slower using the default mass unit (Dalton) in comparison to trial using the “ppm” mass unit. This directly affects the speed of our PSO model over several Comet trials. Figure 3 shows the difference in execution times of PSO with different mass units. In most cases, even
with wider predefined range of \( w \) (see Table II), PSO runs almost three to four times faster using the “ppm” mass unit.

Figure 4.3: Execution times of PSO using different mass units.

Table 4-V: PSMs from Comet based on tolerance windows recommended by PSO, default windows, and Param-Medic windows. The difference reported is with default parameters.
As to the HEK293 dataset, unfortunately, *Param-Medic*, in the current Linux based version of Crux 3.1, cannot find enough information in any file to estimate the window tolerance. This dataset has 24 MS/MS spectra files that have been fed into Comet one by one in this study. Table VI shows the performance of PSO in searching the best parameter value using HEK293 spectra. The matches (PSM) in the table are identified by Comet using both values of tolerance window: 20 ppm (constant) and the one recommended by PSO. The results show that the optimal value of this parameter is hard to keep it constant or fixed for all files. This supports our conclusion and subsequent future work to design spectrum level parameters better. Despite the few extra identified spectra, Table VI shows the capability of PSO in finding better values of this parameter to maximize the matches. PSO suggests better value than the constant value (i.e. 20 ppm) in 20 files of HEK293 dataset, while the other PSO recommendations do not help Comet in finding 6 to 28 more correct identifications in 4 files.

<table>
<thead>
<tr>
<th>File</th>
<th>PSO</th>
<th>20 ppm (Constant)</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Matches</td>
<td>(w)</td>
<td>Matches</td>
</tr>
<tr>
<td>b1906_293T_proteinID_01A_QE3_122212.raw</td>
<td>14711</td>
<td>24.27</td>
<td>14717</td>
</tr>
<tr>
<td>b1922_293T_proteinID_02A_QE3_122212.raw</td>
<td>17131</td>
<td>11.7</td>
<td>17087</td>
</tr>
<tr>
<td>b1923_293T_proteinID_03A_QE3_122212.raw</td>
<td>14697</td>
<td>26.3</td>
<td>14676</td>
</tr>
<tr>
<td>b1924_293T_proteinID_04A_QE3_122212.raw</td>
<td>16765</td>
<td>20</td>
<td>16765</td>
</tr>
<tr>
<td>b1925_293T_proteinID_05A_QE3_122212.raw</td>
<td>16027</td>
<td>42.3</td>
<td>15984</td>
</tr>
<tr>
<td>b1926_293T_proteinID_06A_QE3_122212.raw</td>
<td>15041</td>
<td>41.4</td>
<td>14996</td>
</tr>
<tr>
<td>b1927_293T_proteinID_07A_QE3_122212.raw</td>
<td>17441</td>
<td>23.7</td>
<td>17266</td>
</tr>
<tr>
<td>b1928_293T_proteinID_08A_QE3_122212.raw</td>
<td>14364</td>
<td>13.5</td>
<td>14349</td>
</tr>
<tr>
<td>b1929_293T_proteinID_09A_QE3_122212.raw</td>
<td>19604</td>
<td>19.2</td>
<td>19617</td>
</tr>
<tr>
<td>b1930_293T_proteinID_10A_QE3_122212.raw</td>
<td>15691</td>
<td>14.6</td>
<td>15719</td>
</tr>
<tr>
<td>b1931_293T_proteinID_11A_QE3_122212.raw</td>
<td>19281</td>
<td>9.3</td>
<td>19169</td>
</tr>
<tr>
<td>b1932_293T_proteinID_12A_QE3_122212.raw</td>
<td>14287</td>
<td>27.1</td>
<td>14210</td>
</tr>
</tbody>
</table>
Table 4-VI – continued

<table>
<thead>
<tr>
<th>File</th>
<th>PSO Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1937_293T_proteinID_01B_QE3_122212.raw</td>
<td>16589</td>
</tr>
<tr>
<td>b1938_293T_proteinID_02B_QE3_122212.raw</td>
<td>14277</td>
</tr>
<tr>
<td>b1939_293T_proteinID_03B_QE3_122212.raw</td>
<td>17272</td>
</tr>
<tr>
<td>b1940_293T_proteinID_04B_QE3_122212.raw</td>
<td>15503</td>
</tr>
<tr>
<td>b1941_293T_proteinID_05B_QE3_122212.raw</td>
<td>14825</td>
</tr>
<tr>
<td>b1942_293T_proteinID_06B_QE3_122212.raw</td>
<td>16846</td>
</tr>
<tr>
<td>b1943_293T_proteinID_07B_QE3_122212.raw</td>
<td>13583</td>
</tr>
<tr>
<td>b1944_293T_proteinID_08B_QE3_122212.raw</td>
<td>18353</td>
</tr>
<tr>
<td>b1945_293T_proteinID_09B_QE3_122212.raw</td>
<td>7507</td>
</tr>
<tr>
<td>b1946_293T_proteinID_10B_QE3_122212.raw</td>
<td>19097</td>
</tr>
<tr>
<td>b1947_293T_proteinID_11B_QE3_122212.raw</td>
<td>11946</td>
</tr>
<tr>
<td>b1948_293T_proteinID_12B_QE3_122212.raw</td>
<td>18299</td>
</tr>
</tbody>
</table>

Table VII shows the time in minutes required to find the best value of window tolerance. Besides the encouraging results in Table VI, Table VII also supports the PSO solution to optimize protein search engines since the overhead of parameter searching time could be reasonable.

Table 4-VII: Execution times of PSO using ppm mass unit on HEK293 dataset files.

<table>
<thead>
<tr>
<th>File</th>
<th>PSO Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>b1906_293T_proteinID_01A_QE3_122212.raw</td>
<td>183.00</td>
</tr>
<tr>
<td>b1922_293T_proteinID_02A_QE3_122212.raw</td>
<td>120.00</td>
</tr>
<tr>
<td>b1923_293T_proteinID_03A_QE3_122212.raw</td>
<td>106.80</td>
</tr>
<tr>
<td>b1924_293T_proteinID_04A_QE3_122212.raw</td>
<td>122.40</td>
</tr>
<tr>
<td>b1925_293T_proteinID_05A_QE3_122212.raw</td>
<td>120.60</td>
</tr>
<tr>
<td>b1926_293T_proteinID_06A_QE3_122212.raw</td>
<td>108.84</td>
</tr>
<tr>
<td>b1927_293T_proteinID_07A_QE3_122212.raw</td>
<td>130.00</td>
</tr>
<tr>
<td>b1928_293T_proteinID_08A_QE3_122212.raw</td>
<td>148.20</td>
</tr>
<tr>
<td>b1929_293T_proteinID_09A_QE3_122212.raw</td>
<td>251.40</td>
</tr>
<tr>
<td>b1930_293T_proteinID_10A_QE3_122212.raw</td>
<td>112.80</td>
</tr>
<tr>
<td>b1931_293T_proteinID_11A_QE3_122212.raw</td>
<td>186.60</td>
</tr>
<tr>
<td>b1932_293T_proteinID_12A_QE3_122212.raw</td>
<td>104.40</td>
</tr>
<tr>
<td>b1937_293T_proteinID_01B_QE3_122212.raw</td>
<td>116.40</td>
</tr>
</tbody>
</table>

105
In summary, both the PSO model and the *Param-Medic* are used to optimize the mass tolerance window, which could maximize the quality of the PSMs, thus, boosting the performance of protein search engines. Table VII shows a summary of the features that we use to compare the two solutions. This table could help interested biologists in choosing the appropriate method to control the search engines being used in their labs.

<table>
<thead>
<tr>
<th>Feature</th>
<th>PSO Model</th>
<th>Param-Medic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speed</td>
<td>Slower</td>
<td>Faster</td>
</tr>
<tr>
<td>cover all window types</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Probability of loss of correct matches</td>
<td>Very Low</td>
<td>High</td>
</tr>
<tr>
<td>Always returns with recommendations</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>covers other parameters</td>
<td>No (Future work)</td>
<td>Yes (ion fragment tolerance)</td>
</tr>
<tr>
<td>Platform independent</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Need experience to use</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Could work with other search engines</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Can be affected by the content of input MS/MS file</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Takes the database into account</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

4.6 Related Work

The mass tolerance window (*w*) parameter has a direct impact on the search performance. In the literature, this parameter is usually described as a gate of random matches; the wider the
window tolerance, the higher the possibility of random matches. From another perspective, narrowing the window results in missing many correct matches. In terms of computation time, the wider the window, the slower the searching process and vice versa. This results from an increased number of candidates being compared against the spectra file [11]. For these reasons, adjusting the mass tolerance window parameter is an important optimization problem where there is a lack of solid solutions in the literature.

In trying to maximize the number of final matches of X!Tandem search engine, a scientific optimizer, Taverna, is used in order to find the best set of X!Tandem searching parameters. Running X!Tandem using optimized parameters resulted in a 9% increase in correct PSMs from human datasets compared to the results using default parameters. The running times using optimized parameters exceed the running time using default parameters by around a factor of two in many cases. In addition, the optimization time varied from 3 hours to around 30 hours on a computation cluster. The sequence files have 4,439 real E. coli protein sequences and 89,601 human sequences, respectively. The MS/MS spectra files are generally a few Megabytes [12]. These recent results have extended the work in [13] using the same optimizers, but also using the power of grid computing. In other experiments based on grid search optimization, authors in [14] proposed a trial-and-error method from a set of choices in a grid. In grid searching, several challenges are introduced; the most important ones are the execution time and resources that are consumed in order to find a set of parameters from user-defined parameters. We recently published a paper on the optimization of deep learning algorithms where we demonstrated advantages of PSO over traditional grid searching [15]. Grid searching is inefficient and could sometimes result in poor selection(s) because the set originally has been built with poor choices [15].
Maximizing the final correct PSMs has also been researched not only using parameters of one tool, but also a combination of tools. Searching phase in any protein search engines is usually followed by a filtering phase, or tool. To find the best combination of searching tool and filter, [16] studied the combination of SEQUEST, Mascot and MS Amanda search engines with different post processing approaches; respective score-based filtering, a group-based approach, local false discovery rate, PeptideProphet and Percolator. In different designs, one post-processing event could combine searching results from different tools to maximize coverage, for example, the popular iProphet [17]. Our work in its current form could be utilized in one tool. However, it can be extended in order to leverage the results of multiple tools combinations.

4.7 Conclusion

The peptide mass tolerance window is an influential parameter in protein search engines that could cause loss of correct results if it is unwisely configured. In order to support current search engines in enhancing their coverage, we present a platform independent particle swarm optimizer to find the best value of this parameter. For performance evaluation, the coverage of the Comet search engine was studied using the mass tolerance value recommended by our PSO model, the recently developed Param-Medic tool and the rule of thumb value. For fair comparison, 10 different MS/MS datasets generated by different instruments were randomly selected from the PRIDE public repository and used in our experiments on the same computing machine. In comparison to Param-Medic and the default rule of thumb values, results showed that our PSO increases the number of correct PSMs. Furthermore, PSO is able to handle all of the datasets and find the best tolerance window value, which when used, resulted in improved number of Comet-
identified PSMs. Results also showed that given the current filtering approaches, narrower mass
tolerance windows sometimes improve search engine coverage.

In the future, we will evaluate different sampling techniques of MS/MS spectra and
implement parallel computing in order to speed up our solution.
References


CHAPTER 5

DEEP LEARNING–BASED MSMS SPECTRA REDUCTION IN SUPPORT OF RUNNING MULTIPLE PROTEIN SEARCH ENGINES ON CLOUD


5.1 Summary

The diversity of the available protein search engines with respect to the utilized matching algorithms, the low overlap ratios among their results and the disparity of their coverage encourage the community of proteomics to utilize ensemble solutions of different search engines. The advancing in cloud computing technology and the availability of distributed processing clusters can also provide support to this task. However, data transferring and results’ combining, in this case, could be the major bottleneck. The flood of billions of observed mass spectra, hundreds of Gigabytes or potentially Terabytes of data, could easily cause the congestions, increase the risk of failure, poor performance, add more computations’ cost, and waste available resources. Therefore, in this study, we propose a deep learning model in order to mitigate the traffic over cloud network and, thus reduce the cost of cloud computing. The model, which depends on the top 50 intensities and their m/z values of each spectrum, removes any spectrum which is predicted not to pass the majority voting of the participated search engines. Our results, using four different datasets and three search engines in the first experiment, namely: pFind, Comet and X!Tandem, are promising and promote the investment in deep learning to solve such
type of Big data problems. In the second experiment, Comet, pFind and Crux-Tide are utilized in
the majority voting process on HEK293 dataset of more than a million spectra. The results
further support our hypothesis that deep learning model can be used as a good starting point
towards improving the performance of cloud-based protein search engines.

5.2 Introduction

In bottom up proteomics, the identification of peptides; the basic blocks of proteins, is an
essential task towards protein identifications. Proteins are identified by their peptides represented
by a set of input MSMS spectra. Searching the input spectra against proteins’ database is the
most common and reliable technique currently used in proteomics. Briefly, each MSMS
spectrum is compared against a set of candidate peptides retrieved from a known peptides’
database. Then, each spectrum is paired with a set of candidate peptides which could likely
represent one or more of them. However, the post processing filters usually reject some of them,
sometimes the majority, and report only those PSMs (peptide-to-spectrum matches) of high
probabilities of correctness [1]. The current search engines usually report different results in
terms of the number of PSMs (coverage) or the identified peptides themselves. This scenario is
typical even for the same input of MSMS spectra and databases. This problem of low overlap
ratios among the search engines results can create unnecessary controversial issue in terms of
questioning the results.

The diversity of the available protein search engines with respect to the utilized
algorithms, either in the matching process or in the post-processing, encourages the community
of proteomics to get benefits from combining them. Combining the search results could result in
a better coverage and more accurate identified proteins. However, from computation and
performance perspective, there is a major challenge in running several tools and combining their results either on one machine or on the cloud. The current search engines’ capacities cannot handle even small to medium size data volumes and they take unreasonable time in peptides searching and post-processing the results. This simply indicates how difficult is running several tools on one machine, even with multicores, particularly, if the database and the input observed spectra are beyond the capacity of the machine’s specifications. The advancing in cloud computing technology and the availability of distributed processing clusters could make the running of several tools possible and efficient. The muscles are ready to handle the load if the brain manages that intelligently. Despite the speed of computations on cloud environments, data transferring and results’ integration, in this case, could be the bottleneck. The flood of billions of observed mass spectra, hundreds of Gigabytes or potentially Terabytes, could easily cause the congestions, increase the risk of failure, poor performance, add more computations’ cost, and waste the available resources. Figure 1 below shows typical rather naïve design for majority voting system of three different search engines.

One of the main advantages of combining results of multiple search engines could be in increasing the reliability of the matches (PSMs). The consensus answer of different search engines for a spectrum could increase the probability that this match (PSM) is not a random match. In this paper, we propose a deep learning model-based solution which removes any spectrum, from the cloud-based system, if the spectrum did not pass the majority voting of the participated search engines. This means to mitigate the traffic over the interconnection network of the cloud solution, reduce the required space in each computing node over the cloud, reduce the search time since some the input MSMS spectra are excluded and may also increase the reliability of the final results due to the elimination of some spectra with poorly rated results.
5.3 Background

Improving the accuracy of protein identification is the main goal of the community of proteomics. There are several tools developed in the last decade where reporting more accurate results within reasonable time is the desired goal. However, up to the time of writing of this paper, processing the same input observed spectra and the database by different tools results in different identifications, even the overlaps among them are, most of the time, very few. This indicates that each tool has a different methodology and metrics in peptides deduction, and picking one of them as a search engine of choice is really hard.

One of the best strategies in improving the reliability and the accuracy of peptides deduction systems is to bring the power of different tools together, where a unified system of their results could improve both the coverage and the accuracy of deduction. One concern could be the selection of search engines that should go together in one unified system since it plays a major role in the final results and on the whole system performance [2]. The usefulness of building combiners and how could the results become more robust are discussed and proposed in several researches, for more details, results and discussions one could refer to [3-10]. Regardless the approach of evaluating and reporting the final results in combiner systems, the main concerns
being discussed in this study are the computations [2] and the deluge of data [1] challenges. These are typical challenge issues any big data analysis process may face.

Obviously, given the inability of the current search engines in handling a large volume of spectra and databases, running multiple search engines in one machine is not an option. There are billions of observed spectra that are lined up in the waiting queue, where the limited capabilities of the typical desktop machine are struggling in executing multiple search engines. The advance technology of cloud computing provides an efficient distributed environment where several big data applications utilize the scalable computing resources. Processing the mass spectra search on cloud shows an efficient speed up compared to the serial versions even with some manual effort and multiple virtual machines to run the algorithm as in [11]. The map reduce over Amazon cloud, AWS, is utilized in proteomics where the X!Tandem search engine is improved and transformed into a map reduce application, MR-Tandem [12]. Searching multiple files could be distributed over different nodes over AWS, and the results are combined for post-processing [13] at the end. Other projects that also tried to bring the power of cloud computing include: CPFP, the Central Proteomics Facilities Pipeline [14], hydra [15], and ProteoCloud [16].

In [17] a scalable, efficient, and unified solution of multiple algorithms of peptide deduction is developed over AWS. The key which makes this work different is the ability of running multiple search engines of peptide deduction. The efforts have been spent to support this project either by the cost estimator application or by the online intensive tutorials to train users. This could really add a significant value to the users and researchers in proteomics that have a diversity of backgrounds; computer science, biology, biochemistry, statistics, etc.
As discussed in the introduction, and also in [17], the concern of moving observed spectra to the computation nodes and reporting the results back to the unified algorithms for post-processing should be seriously discussed and taken as an important parameter in evaluating this kind of cloud projects. One of the possible solutions is to reduce the size of the input spectra either via peak picking [18] or whole spectra elimination. In fact, not all spectra are useful in protein identification and the majority is ignored from further processing [19]. There are several reasons behind that, including for example, the search engine methodology, the database, the quality of the spectrometer, etc. From that perspective, eliminating spectra could be useful to mitigate the traffic of the sheer volume over the cloud channels, since not all of them are useful for protein identification, at least from the utilized search engine perspective. Machine learning approaches could be utilized in this context; in [20] a classifier was built in order to eliminate the unidentifiable spectra by SEQUEST search engine, which directly impacts the performance of SEQUEST and the quality of results where the loss of correct and identifiable spectra does not exceed 10%. Another MASCOT-based classifier using manually specified attributes such as the number of peaks, precursor charge, etc., was able to eliminate 62% of the unidentifiable spectra where only 2% or less of the identifiable spectra is lost [21]. In [19] a quality scoring algorithm was developed without any training labeled dataset, but dynamically depending on the initial searching pass, which means more robust algorithms toward different spectra and devices. For the same purpose, a reduction technique is applied over the input spectra based on their precursor charge and the expected fragment ions. Any spectrum falls under the threshold value is totally removed; see [22] for more details.

In the next sections, we discuss our proposed solution using deep learning classifiers. Our model, mainly, depends on the vector of intensities and their m/z (Mass to charge ratio) values.
The encouraging results we got promote the investment in machine learning to make these cloud-based solutions more efficient.

5.4 Methods and Materials

5.4.1 Dataset

In the first experiment, we randomly picked four different public datasets of different species and instruments. We downloaded them from PRIDE repository. Table I shows the properties of the datasets and the downloaded information. We distinguished the human datasets by using different suffix which is related to our datasets indexes. As depicted in table I, each one of the datasets reflects different properties in terms of instrument and the spices, or tissue. This diversity is important for this kind of research.

In the second experiment, we evaluate the machine learning solutions using the same features using HEK293 dataset. This dataset comes in 24 large files. There are more than a million of high resolution spectra generated by Q Exactive device (see Table I). Each file is processed separately in this study. The overall performance of the algorithms is calculated based on the average value and 95% confidence interval. For more information about this large dataset, reader is referred to PRIDE website using the accession number PXD001468.

5.4.2 Deep Learning Algorithm

We conducted all experiments using h2o package which was built under R 3.2.4. In order to get the best results from deep learning algorithms, one should carefully consider the configuration of the deep learning parameters.
There are several parameters to tune including the number of hidden layers and the neurons in each layer. We recently developed a configuration method using Particle Swarm Optimization (PSO) to pick the best parameter values for both number of layers and neurons [23]. In this study, deep learning models are generated using our PSO models.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accession #</th>
<th>Instrument</th>
<th>Tissue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human01</td>
<td>PXD003651</td>
<td>LTQ Orbitrap</td>
<td>prostate epithelium</td>
</tr>
<tr>
<td>Human06</td>
<td>PXD004193</td>
<td>LTQ Orbitrap XL</td>
<td>NA</td>
</tr>
<tr>
<td>Rat</td>
<td>PXD004150</td>
<td>Q Exactive</td>
<td>Brain</td>
</tr>
<tr>
<td>Soybean</td>
<td>PXD001943</td>
<td>LTQ Orbitrap Velos</td>
<td>embryonic axis</td>
</tr>
<tr>
<td>HEK293</td>
<td>PXD001468</td>
<td>Q Exactive</td>
<td>Brain. Kidney</td>
</tr>
</tbody>
</table>

5.4.3 Protein Search Engines

In the first experiment, three semantically different search engines to generate the datasets for machine learning algorithms are utilized in this study. X!tandem [24], Comet and its Percolator [25] and pFind [26] have searched the first four datasets in Table 1 against Uniprot_sprot database.

The diversity of dataset and search engines is an important issue in any data analysis research. For this reason, another evaluation experiment is also conducted using different spectra dataset, protein database and different voters (i.e. search engines). In this second experiment, Crux-Tide [25], Comet and pFind search engines have searched the HEK293 dataset in Table 1 against Human database; ipi.Human.v3.87. The Percolator algorithm is used to filter the results of Crux-Tide and Comet matches.
5.4.4 Datasets for Machine Learning

We combined the search results for each dataset from the aforementioned search engines. Each spectrum is labeled as “1” or “0”. We used “1” to label the spectrum which has consensus identification. It simply means that at least two search engines have assigned the same peptide of the spectrum. We used “0” if the spectrum does not pass the consensus identification. For example, after searching the database by three search engines (X!Tandem, Comet and pFind in the first experiment) to identify the spectrum of precursor 415.7, Comet and X!Tandem reported “TSNGENGR” and pFind did not identify the spectrum. In this case, the spectrum has passed the majority voting (consensus) conditions and labeled as “1”. For another spectrum of precursor 557.63, only Comet identified the spectrum while X!Tandem and pFind did not. In this case, the spectrum was labeled as “0” since it did not pass the majority voting process. As a third example, the spectrum of precursor 708.89 is identified differently by the three search engines. Each one of them is assigned different peptide which the spectrum could represent. In this case, the spectrum was also labeled as “0”.

In order to build machine learning models, each spectrum is represented by a vector of 100 values; the highest 50 peaks and their m/z (mass to charge ratio) values. We chose the top 50 since we do have this amount of peaks most of the time. Different ionization methods could produce spectra of few peaks (sometimes less than a hundred) up to thousands of them. Moreover, building simple models is important in this case since the model will be centralized and play a viable role in reducing the data transferred over the cloud interconnection network. One more reason is that some of the current search engines already use top peaks in their searching, and using the top 50 peaks in our model could be an advantage to utilize the same data and eliminate any preprocessing to test the spectrum using our model.
As for the search parameters, we used the same parameters for Crux-Tide, Comet, X!Tandem and pFind search engines. Table II shows the datasets that were used to build the machine learning models, and their sizes. The size of HEK293 is based on Comet search engine v 3.1.

<table>
<thead>
<tr>
<th>#</th>
<th>Dataset Name</th>
<th>Dataset size (# of Spectra)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Human01</td>
<td>10,115</td>
</tr>
<tr>
<td>2</td>
<td>Human06</td>
<td>8,771</td>
</tr>
<tr>
<td>3</td>
<td>Rat</td>
<td>4,022</td>
</tr>
<tr>
<td>4</td>
<td>Soybean</td>
<td>15,412</td>
</tr>
<tr>
<td>5</td>
<td>HEK293</td>
<td>1,119,064</td>
</tr>
</tbody>
</table>

5.4.5 Models Evaluation

For the evaluation process, each dataset is randomly shuffled then divided into four folds. We used 75% (3 folds) of the dataset for training and 25% (1 fold) for testing datasets. We used three metrics in evaluating the classifiers: the overall accuracy, sensitivity, and specificity. The sensitivity in this study means how many times the classifier predicts “1” where the real label is “1”. The specificity shows how many times the classifier predicts “0” where the real label is “0”. For Cloud based solution of different search engines, sensitivity means how many spectra are correctly sent to the computing nodes over the cloud network, and similarly, specificity means how many spectra are correctly removed and have not sent to the computing nodes over the cloud network. The ideal case is to have high sensitivity and specificity at the same time. However, in real life applications, this combined goal is not easy to achieve.
For comparisons and to show the value of deep learning over other algorithms in this context, the same inputs (Folds) were processed using two of the commonly used classification algorithms in machine learning; namely Support Vector Machine (SVM) and Logistic Regression. We show simple comparisons between the deep learning model performance and their performance.

5.5 Results and Discussion

5.5.1 The First Experiment

Figure 2 depicts the performance of machine learning models in Human01 dataset using 95% confidence interval. The classifier of deep learning is able to correctly predict about 86%, on average, of the spectra that should be sent to the search engines over the cloud and only 52% of them that should not be transferred. This simply means that we eliminated more than half of the unnecessary load where we lose, on average, 14% of spectra that could have consensus answers. Using 95% confidence interval, SVM and Logistic Regression can remove more unnecessary load from the system, however, around 30% of the desired spectra will be filtered out incorrectly. From the perspective of Securing the desired spectra, deep learning model would be the choice since it shows high sensitivity compared to the other two models.
Figure 3 shows the performance of the models on Human 06 data set using 95% confidence interval. Similar to the deep learning model on Human01 data set, the deep learning model was able to reduce the unnecessary load into more than half where the loss does not exceed 14% of the good spectra that should be transferred. The shallow algorithms show encouraging results in terms of reducing the unnecessary spectra load but at the expense of losing more good spectra. This may not be desirable as the loss is around 25% of good spectra.

Figure 4 shows a better performance of all models on the Rat data set. Using 95% confidence interval, the deep learning model was able to correctly classify around 90% of both the necessary and unnecessary loads. Clearly, here the other models also show good and comparable results to deep learning.
Using Soybean dataset, the deep learning model was able to exclude 51% of the unnecessary load while 13%, on average and using 95% confidence interval, of the necessary load is lost, see Figure 5. The results of other classification algorithms show that the loss ratio of necessary load (desired spectra) which is around 30% makes the deep learning models better. This is at least from the perspective of securing good spectra.
5.5.2 The Second Experiment

This section discusses the results of using different voting systems which consist of Crux-Tide, Comet, and pFind search engines on HEK293 dataset and human database. The dataset is large, where more than a million of high-resolution spectra are searched against the human database. Figure 5.6 (A-X) shows the performance of deep learning models and other shallow learning algorithms on the dataset files.

(A) b1906_293T_proteinID_01A_QE3_122212.raw

(B) b1922_293T_proteinID_02A_QE3_122212.raw
Based on the above encouraging results, we conjecture that deep learning algorithms are efficient in reducing the unnecessary load on cloud based solutions in bottom up proteomics. Certainly, we need more improvements and experiments to reduce more unnecessary loads. However, given the flood of billions of MSMS spectra to the computing nodes that host the search engines, reducing the unnecessary load into half is clearly significant, especially when around 83% -85% of the correct and identifiable spectra are transferred to the search engines. For
the other algorithms, SVM and Logistic Regression, the higher ratios of loss of necessary load (desired spectra) could be a concern. The advantage of the deep learning in this case is the sensitivity compared to SVM and Logistic Regression. Moreover, in the above experiments, it is clear that other algorithms show comparable results, as in Rat dataset, because of the size of the dataset. In this context, usually search engines deal with millions of input spectra [27] which could make the deep learning algorithms the biologist choice for their better performance in supporting voting-based systems.

Regarding the predicting time, which is important in these kinds of models and systems, our deep learning model can return predictions of thousands of MSMS in testing datasets within few seconds. This indicates that our deep learning model is efficient in terms of the time and of processing data and the accuracy.

5.6 Conclusion

We developed a deep learning model in support of running multiple search engines on the cloud. Combining multiple search engines’ results has several advantages including, but not limited to, the increase of the trustiness or the accuracy in the final results or PSMs (Peptide-to-Spectrum Matches) and the improvement of the coverage (the final number of identified MSMS spectra). Typically, in running multiple search engines for majority voting, we are looking for those spectra where multiple search engines agree on peptides that they could represent. Cloud-based bottom up proteomics could be an efficient solution where we can run multiple search engines on distributed computing resources. However, the size of input MSMS spectra, which could reach terabytes, could be a bottleneck. The input Terabytes of MSMS spectra should be transferred to each search engine over the cloud interconnection network. This simply means a
large data over the interconnection network and a burden on the nodes themselves that host the search engines. In majority voting process, we noticed that not all MSMS spectra are necessary to be transferred since different search engines did not agree on one answer or some of the spectra could not be identified by the majority of the utilized search engines.

Our deep learning model is trained over the top 50 spectra peaks and their m/z values only in order to eliminate those unnecessary MSMS spectra for the ensemble solution. Our deep learning model can reduce the unnecessary load into almost a half while losing only around 15%-17%, on average, of the necessary ones. Support Vector Machine and Logistic Regression models can reduce more of the unnecessary load than the deep learning model, but eliminating around 30%-50% of the necessary ones too. This could make their performance unacceptable, compared to deep learning model, especially if the goal is to secure more good spectra. In future, we will evaluate ensemble machine learning solutions and add more datasets for the evaluation process.
References


6.1 Summary

Several peptides search engines have been developed in the recent decades. Most of the time and for the same inputs, different search engines’ result in different peptides were identified, which can confuse the stakeholders in the field of proteomics. The massive amount of generated spectra by high throughput spectrometers adds another challenge which handicaps the current search engines. This motivates the researchers to evaluate the combination of several search engines. Several studies provided ensemble solutions over shared and distributed computing environments for reliable results. However, the massive amount of MS/MS spectra is a cumbersome traffic over the systems’ networks. This issue directly impacts the searching performance and also adds unnecessary extra costs (computing, storage, network traffic) if cloud cluster is being used. The main question of this paper is: Can we build a central MS/MS spectra preprocessing for semantically different protein search engines? We evaluate different statistical reduction techniques using four popular protein search engines. In order to fairly evaluate the results, we build ground truth unanimous-based datasets for two different species; yeast and human. Our techniques result in significant peak reduction, where only around 30% of the
spectra peaks are enough to report reliable identifications from the used search engines in this study.

6.2 Introduction

In mass spectrometry based proteomics, inferring the peptides is an essential task toward identifying the proteins in the chemical sample. Searching the MS/MS spectra against a predefined proteins database is the most common technique being used nowadays in shotgun proteomics [1]. The process starts from the biological samples where the unknown proteins are digested using enzymes, Trypsin for example, into peptides, yet they are unidentified. Those peptides are fed into high throughput mass spectrometers where a massive amount of spectra is generated in a short time. For example, more than 24 Gigabytes of compressed spectra are produced from a Thermo Fusion device in one day [2]. Quadrupole-Orbitrap instrument generates around 14 Gigabytes of raw spectra per day [3]. One Gigabyte of raw spectra is generated per hour by Q-T instrument and LC MS/MS [4]. After that, the computational challenge starts in order to identify the best matches of spectra by searching another massive size of a protein database, and finally identifying the peptides into proteins.

The advanced technology of spectrometers demands developing high throughput systems for peptide-to-spectrum match (PSM) process. In addition, the low overlap ratios among the outputs of several popular search engines for the same input, [5], creates a challenge to the proteomics research community in terms of which tools to trust for their research. Building distributed PSM ensemble systems using several search engines could increase the degree of trustworthiness in peptides’ deduction results. However, there are several challenges that should also be addressed. The massive size of those spectra could be the major concern in building
distributed systems. We evaluate several peaks’ reduction techniques as possible centralized solutions to reduce the deluge of spectra over the ensembles’ solutions networks. Due to the lack of benchmarks or reference databases, we propose building a unanimous-based dataset as a reference dataset to evaluate various reduction techniques. The reference datasets have spectra of different properties; precursor values, and number of peaks, which indicate that one can trust this unanimous-based dataset to evaluate similar experiments as a solid assumption and as a starting point for further experiments. Our results of this paper are applicable to a wide variety of computational biologists dealing with high throughput mass-spectra and protein identifications.

6.3 Background

Googling about proteins search engines results in many hits; these could be mainly categorized as: Commercial such as SEQUEST [6], MASCOT [7], PEAKS [8], and freeware or open source, such as Crux [9], OMSSA [10], X!TANDEM [11]. Answering the question about the tools of choice warrants deep analysis and experiments to establish, at least, some rules of thumb. Obviously, biologists look for the coverage (the percentage of identified spectra), the fastest tool, and of course the accuracy. Therefore, one solution could be a combination of different tools [12-14]. Consensus-based proteomics, where the spectrum is identified by consensus scoring or voting, could be used as well to increase the degree of trustworthiness and the reliability in peptides’ deduction results [15-16]. Another alternative is the majority voting systems [17].

Consequently, in ensemble systems, while increasing the reliability of results, the load of running several tools is spread over distributed processes and managed by the power of high performance computing. However, broadly speaking, building such PSM ensemble systems
requires paying attention to different factors which possibly burden the entire system and could lead to failure. Figure 1 shows the abstract components of a typical and rather naïve proposed distributed ensemble system using the search engines of this study. Typically, in designing distributed systems, data distribution and traffic over the interconnection networks is always a concern. In our case, the observed spectra should be sent to all distributed processes that are running search engines, and after that, collect the results of matching processes in order to calculate the votes, which simply results in a data flood. We therefore evaluate the usefulness of spectra peaks reduction techniques; some of them are discussed in [18], in order to build a centralized and unified reduction technique.

![Diagram](image)

Figure 6.1: Major components of a naive distributed computing solution design for the majority voting mass spectra interpretation system.

6.4 Methods

6.4.1 Proteins Search Engines

we use four popular search engines currently freely available and downloadable, and also used in several prominent proteomics and related research work, namely, Crux-Tide, OMSSA, PFind [19], and X!Tandem. For the OMSSA and X!Tandem, we use the versions implemented in
PeptideShaker [20]. To the best of our knowledge, ours is the first work to evaluate them cohesively in one experiment. Crux-Tide utilizes cross correlation in order to rank candidate peptides, which the input observed spectrum possibly represents. The top candidates are then filtered by percolator [21], which reports the best match given the inputs. X!Tandem depends on hypergeometric score in matching observed spectra against the theoretical ones, where the score, namely hyperscore, includes the number of y- and b-ions. OMSSA depends on Poisson probability distribution to calculate the E-values for each spectrum scores which were calculated based on the number of matches of shared peaks between the input spectrum and the retrieved candidates. Regarding pFind tool, it depends on an advanced approach to strengthen the dot product scores by utilizing the kernel trick, namely of including the relative information among spectrum fragments ions.

6.4.2 Datasets

Three different datasets are used to make sure that our inferences and conclusions are not data-dependent. The first dataset is a yeast dataset which was used in evaluating the accuracy of percolator towards correctness of peptide-to-spectrum-matches [21]. The second dataset is a human data generated from LTQ instrument and can be downloaded from PeptideAtlas repository under accession number PAe001648. The FASTA database file is also available under the same accession number. The third dataset, HEK293, was used to evaluate the Top X% reduction using unanimous dataset generated from Comet and Crux-Tide. This dataset is available on PRIDE (https://www.ebi.ac.uk/pride/archive/projects/PXD001468).
6.4.3 Building Unanimous-Based Dataset

We depend on the unanimous method where all the PSM tools under study agree on the same answer of the input query. The spectrum must have 4 similar identifications by four different tools to be involved. This might build a smaller albeit a more trusted dataset, making our reference dataset more sound and solid.

6.4.4 Peaks Reduction

Taking into consideration the importance of the most intensive peaks[22], we implement three different methods of sampling the peaks with the aim of reducing their numbers in every input spectrum; namely uniform cut-off, Top X%, and multistage reduction using a hybrid approach of strata (or regions) and Top X%. We also consider the possibility of using simple random sampling (SRS) as a reduction technique since it could be faster and easier to implement than the other three. Assuming a uniform distribution in any spectrum, the probability of any random bin (peak) that belongs to the interval \([\text{min (m/z)}, \text{max (m/z)}]\) can be expressed as:

\[
P(\text{PEAK} = \text{peak}) = \frac{1}{\text{Max} \frac{m}{z} - \text{Min} \frac{m}{z}} \ldots \ldots (1)
\]

where \(m/z\) represents mass-to-charge ratio. In order to judge the deduction capability of each search engine using reduced peaks spectra, we use the accuracy value in equation 2, where \((S, P)\) represents spectrum-peptide pair.

\[
\frac{\text{Count(Searching Results(S,P)} \cap \text{Unanimous(S,P))}}{\text{Count(\text{Unanimous(S,P))}}} \ldots \ldots (2)
\]
6.5 Results and Discussion

6.5.1 Unanimous Dataset

The parameters of the searching phase are unified in all the search engines mentioned above. Table I shows the user-defined parameters and their values used in our study. Table II shows that out of 35236 observed yeast spectra and 90746 spectra, only 5253 of yeast spectra and 9754 of human spectra pass the unanimous test. This is the typical case in proteomics and this is why we need more solid and reliable solutions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missed Cleavage</td>
<td>2</td>
</tr>
<tr>
<td>Precursor Tolerance</td>
<td>3.0 Da</td>
</tr>
<tr>
<td>Fragment Tolerance</td>
<td>0.2 Da</td>
</tr>
<tr>
<td>Enzyme</td>
<td>Trypsin</td>
</tr>
<tr>
<td>Mass type</td>
<td>Mono</td>
</tr>
<tr>
<td>FDR</td>
<td>≤ 1%</td>
</tr>
</tbody>
</table>

Table 6-II: The identified and observed spectra and the unanimous process.

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Identified Spectra (Yeast Dataset)</th>
<th>Identified Spectra (Human Dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crux-Tide</td>
<td>10106</td>
<td>17544</td>
</tr>
<tr>
<td>X!Tandem</td>
<td>35066</td>
<td>11744</td>
</tr>
<tr>
<td>pFind</td>
<td>31406</td>
<td>11659</td>
</tr>
<tr>
<td>OMSSA</td>
<td>19972</td>
<td>11425</td>
</tr>
<tr>
<td>Unanimous Spectra</td>
<td>5253</td>
<td>9754</td>
</tr>
</tbody>
</table>
In order to validate the heterogeneity of the properties of the unanimous spectra and the peptides they represent, we study and analyze the unanimous spectra with respect to all the input spectra. Similarly, we also study the identified peptides of the unanimous spectra with respect to all peptides in the input database. Figure 2 shows the precursor values for Yeast dataset, and Human dataset, respectively. All observed spectra precursor values are represented in green dots, but those elected as unanimous spectra are shaded by black color. This easily explains that the observed spectra do not represent any special cases with respect to precursor values.

6.5.2 Evaluation of Uniform Cut-Off Reduction Technique

Figure 3 shows the distribution of spectra sizes after applying the uniform cut-off on the Yeast and the Human dataset. While there are several spectra that have not been reduced at all, some of them have been reduced to less than half in the Yeast dataset, and to around 10% of their original number of peaks such as those in the Human dataset. On an average, 70% of peaks are retained in spectra of the Yeast dataset, whereas only 28% are retained in the Human spectra dataset. It is clear that the majority of human spectra are even represented by less than 20% of
the original peaks. Table III shows the accuracy values of each search engine after searching the same input database using the newly reduced-peaks spectra. We noticed that X!Tandem results were not affected at all by excluding those peaks in both dataset spectra. Crux-Tide and pFind also report very high accuracy values, almost 100% for both the datasets. OMSSA reports 87.6% of yeast spectra correctly and did almost 100% on the human spectra. Compared to others, OMSSA reports an accuracy of 87.6% which could be unacceptable given the reduction factor of only 30% on an average (i.e., 70% of peaks still exist), but it successfully handled the human spectra where they have only 28% of their retained peaks on an average.

![Figure 6.3: Peaks retained after applying uniform cut-off.](image)

Table 6-III: Comparisons of the accuracy values of different tools and datasets.

<table>
<thead>
<tr>
<th>Search Engine</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Yeast</td>
</tr>
<tr>
<td>Crux-Tide</td>
<td>98.6%</td>
</tr>
<tr>
<td>pFind</td>
<td>99.9%</td>
</tr>
<tr>
<td>OMSSA</td>
<td>87.6%</td>
</tr>
<tr>
<td>X!Tandem</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
6.5.3 Evaluation of Top X% Reduction Technique

In Top X%, the intensities are sorted in a descending order, and only the highest X% of them are retained. We use 9 values of X in order to study and represent visually the behavior of the search engines in handling reduced peaks spectra, where \( X \in \{10, 20, 30, 40, 50, 60, 70, 80, 90\} \). Figures 4 and 5 show the accuracy calculated from the results of searching the reduced-peaks obtained using top X%. In both datasets, X!Tandem and pFind search engines report almost 100% of the input reduced spectra even with the lowest peaks retaining percentages. OMSSA is in the third position and the Crux-Tide always reported the lowest percentage of correct answers. However, in the Human dataset, all of them report a high percentage of correct answers even with less than 20% of retained peaks per spectrum. We use different scale on the Y-axis in Figure 5 to show the minor differences among the search engines. As depicted in Figure 6, this explains the reasons behind these results, the number of peaks retained in the Yeast spectra are very few compared to those retained in the Human spectra when \( X=10 \) (the reduction factor = 90%). This is why the algorithms of those tools still can find enough Human useful peaks for peptide deduction.
Figure 6.4: Comparison of accuracy values of the four search engines results after applying Top X% on yeast spectra.

Figure 6.5: Comparison of accuracy values of the four search engines results after applying Top X% on human spectra.
6.5.4 The Evaluation of Multistage Reduction Techniques

While the Top X% reduction technique of the previous section looks to peaks horizontally, in this section, the vertical view will contribute to building the representative spectrum. Figures 7 and 8 show the accuracy values of searching reduced yeast and human spectra, respectively. We also show the number of retained peaks after applying Top 10% sampling on each region of the nine regions in Figure 9.

Figure 6.6: Retained peaks after applying Top 10% on Yeast and Human spectra.

Figure 6.7: Comparison of accuracy values of the four search engines results after applying multistage reduction on Yeast spectra.
Figure 6.8: Comparison of accuracy values of the four search engines results after applying multistage reduction on Human spectra.

Figure 6.9: Retained peaks after applying Multistage and Top 10% on Yeast and Human spectra.

6.5.5 Evaluating a Simple Random Sampling (SRS) as a Reduction Technique

In SRS, probability based sampling methods, given a spectrum which constitutes K peaks, each peak has the probability \( P(pi) = 1/K \) to be selected as a representative peak. Under the assumption that the majority of any given spectrum peaks are not interpreted as real ions, but simply noise, this gives a very simple theoretical clue that random sampling should result in poor
deduction results. More formally, if $\alpha$ out of $K$ are noise peaks, $\beta$ out of $K$ are useful peaks, and $\alpha \gg \beta$, then $P(pi)$ is dominated by $\alpha$ value. It is rather easy to conclude that the accuracy values of our experiment will follow $\lim_{k \to K} \frac{F(k)}{F(K)} = 1$, which means the larger the sample size, the more accurate the deduction results. The experimental results show conformance to our theoretical analysis, Figures 10 and 11.

![Figure 6.10: Comparison of accuracy values of the four search engines results after applying random sampling on Yeast spectra.](image1)

![Figure 6.11: A comparison of accuracy values of the four search engines results after applying random sampling on Human spectra.](image2)
6.5.6 Evaluating Top X% on Different Dataset and Search Engines

In this section, Comet and Crux-Tide are used together as an ensemble solution. In this experiment, we build the unanimous datasets and the searching has been done using “ppm” window type of value 20. We use \texttt{fragment\_bin\_tol = 0.02} for Comet. Other parameters were kept at default values. We show the performance of Top X% reduction using high resolution dataset downloaded from PRIDE; HEK293 dataset, https://www.ebi.ac.uk/pride/archive/projects/PXD001468. We apply Top X% on spectra peaks after removing zero peaks. Figures 6.12 (A-X) shows the performance of Comet and Crux-Tide search engines on reduced peaks spectra compared to their performance on original peaks.

(A) b1906_293T_proteinID_01A_QE3_122212.raw

(B) b1922_293T_proteinID_02A_QE3_122212.raw
Figure 12 (A-X) show that the results conform the results in the previous sections, where 30-40% of the peaks are enough for those search engines to identify around 98%, on average, of the peptides as if the whole spectra are used in the searching phase.

6.6 Conclusion

Building a PSM ensemble system could be a good solution to increase the dependability on the searching results. However, the flood of mass spectra could prevent even the distributed
high performance systems in searching them. We showed that sampling might be used as a centralized reduction technique where each spectrum can be represented by few peaks, yet useful. Despite the differences among the search engines in preprocessing and ranking the spectra, the different accuracy values per sampling method, and the diversity of the datasets in this study, we found that around 30% of top peaks are enough for all the four search engines (namely Crux-Tide, X!Tandem, pFind, Comet and OMSSA) to deduce the correct peptides. Of course, we were able to judge the sampling methods after we built a ground truth datasets of two different species to be used as reference datasets.

Given the massive amount of observed spectra and semantically different tools, reducing the size into around 30% is really a significant improvement. A flood of 300 GB of mass spectra is a serious challenge, but certainly much better and easier to handle than say a flood of 1 TB.

In terms of sampling time, serially, using Core i3 processor and 4 GB RAM, our simple R script can build a new ms2 file of \(10^4\) observed spectra in 66 seconds. However, we should mention that, in most observed spectra cleaning, filtering, or sampling where the other spectra do not contribute in the process, the dependency between any two randomly selected spectra \(S_i\) and \(S_j\) is zero. This means that it is rather easy to build a scalable parallel solution.
References


CHAPTER 7

CONCLUSIONS AND FUTURE WORK

Despite the richness of research in the literature of protein identification and the availability of a variety of tools, there is still a need for solid and high throughput protein search engines, especially because of the advances in the technology of mass spectrometers which generate a large amount of spectra in short time.

In this dissertation, the main goal is to improve the capacity of the currently available search engines, especially those that are open source search engines and inspired by SEQUEST searching style, for example Crux-Tide. The main focus is on the indexing and searching phases. This dissertation is made up of six significant contributions in improving the performance of protein search engines. The first contribution is the improvement of target/decoy databases indexing. Besides the parallel threads, a clever hashing of peptide’s masses is utilized where each group of peptides have all targets and decoys needed in decoys’ generation and validation. Thus, each thread independently works and runs using also small sizes of containers which enhances the performance of the parallel solution. The parallel version can index the same FASTA file by one third of the serial time using only 4 threads and almost one fourth of serial time using 8 threads. Besides the parallelization, part of the serial code was redesigned so that the memory consumption becomes more efficient. The parallel version can index the same files using around two-third of the memory space that the serial version consumes. Further investigations and experiments are needed to improve the scalability. The improved parallel version is publicly available on https://github.com/wmuwiselab/Protien_HPC.
The second contribution is to reduce the searching space of scoring functions. Deep learning model is introduced in the context of building MS/MS filters. This study supports the quality of search engine results and performance. Ten different supervised machine learning algorithms were compared using different configurations. The comparisons are conducted using 9 shallow learning algorithms with different configurations against deep learning models where the Particle Swarm Optimization controls the deep learning parameters, namely the numbers of layers and neurons. Simple set of features is used in building the models which includes some of those already evaluated in literature and others are new. The new ones can be easily extracted from spectra peaks. Using 11 different datasets generated from two different search engines, different instruments, different species and different sizes, results showed experimental evidence that deep learning models are powerful in filtering MS/MS spectra. They also showed that the feature list is significant where other shallow learning algorithms showed encouraging results in filtering the MS/MS spectra. Those features are very simple and can be extracted in a short and consistent time. Deep learning provided the best overall results with respect to the highest sensitivity -- very important in identifying low concentrations of proteins or those more difficult to identify. Regarding the prediction time, for all models, it does not exceed few seconds on all testing datasets. In the future, as a lesson of this work, the focus will be on deep learning, SVM, Neural Networks, and Random Forest in trying to design more efficient filters where training and testing datasets have millions of observed spectra.

The third contribution is to control the searching space and maximize the number of identified spectra. The peptide mass tolerance window is an influential parameter in protein search engines that could cause loss of correct results if it is unwisely configured. In order to support current search engines in enhancing their coverage, platform independent particle swarm
optimizer is presented to find the best value of this parameter. For performance evaluation, the coverage of the Comet search engine was studied using the mass tolerance value recommended by our PSO model, the recently developed \textit{Param-Medic} tool and the rule of thumb value. For fair comparison, 10 different MS/MS datasets generated by different instruments were randomly selected from the PRIDE public repository and used in our experiments on the same computing machine. In comparison to \textit{Param-Medic} and the default rule of thumb values, results showed that our PSO increases the number of correct PSMs. Furthermore, PSO is able to handle all of the datasets and find the best tolerance window value, which when used, resulted in improved number of Comet-identified PSMs. Results also showed that given the current filtering approaches, narrower mass tolerance windows sometimes improve search engine coverage. In the future, we will evaluate different sampling techniques of MS/MS spectra and implement parallel computing in order to speed up our solution.

The fourth and fifth contributions focus on the running of multiple search engines over distributed environments, for example in-house computing clusters or cloud. Combining multiple search engines’ results has several advantages including, but not limited to, the increase of the trustiness or the accuracy in the final results or PSMs (Peptide-to-Spectrum Matches) and the improvement of the coverage (the final number of identified MSMS spectra). Cloud-based bottom up proteomics could be an efficient solution where we can run multiple search engines on distributed computing resources. However, the size of input MSMS spectra, which could reach Terabytes, could be a bottleneck. The input Terabytes of MSMS spectra should be transferred to each search engine over the cloud interconnection network. This simply means a large data over the interconnection network and a burden on the nodes themselves that host the search engines.
The assumption behind the fourth contribution is to build ensemble solution of different search engines and the peptide deduction is based on the majority voting process. In majority voting process, we noticed that not all MSMS spectra are necessary to be transferred since different search engines did not agree on one answer or some of the spectra could not be identified by the majority of the utilized search engines. A deep learning model is trained over the top 50 spectra peaks and their m/z values only in order to eliminate those unnecessary MSMS spectra for the ensemble solution. The deep learning model can reduce the unnecessary load into almost a half while losing only around 15-17%, on average, of the necessary ones. Support Vector Machine and Logistic Regression models can reduce more of the unnecessary load than the deep learning model, but eliminating around 30-50% of the necessary ones too. This could make their performance unacceptable, compared to deep learning model, if the goal is to secure more good spectra. In future, we will evaluate ensemble machine learning solutions and add more datasets for the evaluation process.

In the fifth contribution, results showed that sampling might be used as a centralized reduction technique where each spectrum can be represented by few peaks, yet useful. Despite the differences among the search engines in preprocessing and ranking the spectra, the different accuracy values per sampling method and the diversity of the datasets in this study, we found that around 30-40% of top peaks are enough to deduce the correct peptides for the five search engines being evaluated (namely Comet, Crux-Tide, X!Tandem, pFind and OMSSA). Of course, we were able to judge the sampling methods after we built a ground truth datasets of different species to be used as reference datasets. Given the massive amount of observed spectra and semantically different tools, reducing the size into around 30-40% is really a significant
improvement. A flood of 300 GB of mass spectra is a serious challenge, but certainly much better and easier to handle than say a flood of 1 TB.

In terms of sampling time, serially, using Core i3 processor and 4 GB RAM, our simple R script can build a new MS2 file of 104 observed spectra in a minute. However, we should mention that, in most observed spectra cleaning, filtering, or sampling where the other spectra do not contribute in the process, the dependency between any two randomly selected spectra $S_i$ and $S_j$ is zero. This means that it is rather easy to build a scalable parallel solution.