To the Graduate Council:

I am submitting herewith a dissertation written by Bhanu Prasad Rekepalli entitled “Automated Genome-Wide Protein Domain Exploration.” I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Gregory D. Peterson, Major Professor

We have read this dissertation
and recommend its acceptance:

Igor Jouline

Michael Berry

Ethan Farquhar

Accepted for the Council:

Carolyn R. Hodges
Vice Provost and Dean of
the Graduate School

(Original signatures are on file with official student records.)

Automated Genome-Wide Protein Domain Exploration

A Dissertation
Presented for the
Doctor of Philosophy
Degree

The University of Tennessee, Knoxville

Bhanu Prasad Rekapalli
December 2007

Copyright © 2007

Bhanu Rekapalli

Dedicated to my mother Lakshmi Sulochana Vemuri and my father Subba Rao Rekapalli

Acknowledgements

I would like to sincerely acknowledge my advisor Dr. Gregory Peterson for his kind, encouraging, and helpful supervision throughout the course of my research and studies in The University of Tennessee, Knoxville. I would like to thank my dissertation committee members, Dr. Ethan Farquhar, Dr. Michael Berry, and Dr. Igor Jouline for reviewing and directing my dissertation.

I sincerely thank Dr. Igor Jouline for guidance and support for picking my topic of interest. Working in Igor's Lab generated this idea of dissertation without which the dissertation would not have been existed. I am indebted to my nice and awesome boss John Rose of Office of information Technology, for support and encouragement provided by him for this dissertation. I would like to thank Dr. Luke Ulrich, postdoc in Igor's lab for guiding me during the course of my research.

I am grateful to all my friends who helped in preparation of this dissertation. Special thanks to my best friends Vamsi Nellore, Dr. Vivek Rudrapatna and John Roelofs for giving moral and great support in hard times. Special thanks to my lab members and friends Davi Ortega, Kirill Borziak, Harold Shanafield, and Brian Cantwell for helping me with the biology part of my research.

I would like to thank The University of Tennessee for giving me the opportunity for the Ph.D. program. It’s my pleasure to thank my graduate and undergraduate professors who laid a good foundation for my dissertation work.

Last but most important, I would like to thank whole heartedly, my parents Lakshmi Sulochana Vemuri and Subba Rao Rekapalli and my sister Dr. Deepthi Rekapalli who supported, loved, helped and cared for me throughout my life. I am dedicating this dissertation to them.

Abstract

Exploiting the exponentially growing genomics and proteomics data requires high quality, automated analysis. Protein domain modeling is a key area of molecular biology as it unravels the mysteries of evolution, protein structures, and protein functions. A plethora of sequences exist in protein databases with incomplete domain knowledge. Hence this research explores automated bioinformatics tools for faster protein domain analysis. Automated tool chains described in this dissertation generate new protein domain models thus enabling more effective genome-wide protein domain analysis. To validate the new tool chains, the *Shewanella oneidensis* and *Escherichia coli* genomes were processed, resulting in a new peptide domain database, detection of poor domain models, and identification of likely new domains. The automated tool chains will require months or years to model a small genome when executing on a single workstation. Therefore the dissertation investigates approaches with grid computing and parallel processing to significantly accelerate these bioinformatics tool chains.

Table of Contents

Chapter One ii
1.1 Introduction 1
1.2 Biology Overview 3
1.3 Problem Overview and Motivation 9
1.3.1 Problem 9
1.3.2 Algorithm 11
1.3.3 Challenge 12
1.4 Hardware Architectures Overview 13
1.5 Scope of Dissertation 19
Chapter Two 21
Literature Review 21
2.1 Biological Background 21
2.1.1 Sequence Alignments 22
2.2 Pair-Wise Sequence Alignment Algorithms and Tools 26
2.2.1 BLAST Algorithm 27
2.2.2 BLAST Suite 28
2.2.3 PSI-BLAST 31
2.3 Multiple Sequence Alignment Algorithms 32
2.3.1 CLUSTALW and MUSCLE 34
2.4 Profile Hidden Markov Models and Protein Domain Identification 36
2.4.1 HMMER Suite 37
2.4.2 Domain Identification Tools 38
2.5 Secondary Protein Structure Predictions 42
2.6 Algorithmic and Architectural Accelerators of BLAST and HMMER 44
2.6.1 Algorithmic Speedups 45
2.6.2 Architectural Speedups 47
Chapter Three 50
Automated Tool Chain Design 50
3.1 Domain Identification Automated Tool chain (DIAT) 52
3.2 Domain Verification Automated Tool chain (DVAT) 63
3.3 Domain Discovery Automated Tool chain (DDAT) 67
3.4 PepDomDB Database 73
3.5 Domain Model Verification 73
Chapter Four 75
New Domain Model Results 75
4.1 Test-bench Files 78
4.2 DIAT Results 79
4.3 DVAT Results 83
4.4 DDAT Results 93
4.5 Domain Model Verification Results 96
Chapter Five 99
Computational Results 99
5.1 Architectural Assessment 101

5.2 Computation Times for Shewanella and E.coli Genome-Wide Domain Modeling 104
5.2 Multicore Architectures and Threading 108
5.3 Validation of DIAT 110
5.4 Solved Programming Challenges 111
5.5 Job Mapping and Distribution 112
5.5.1 PSI-BLAST Job Scheduler 114
5.5.2 HMMER Job Scheduler 116
Chapter Six 123
Conclusions and Future work 123
References 128
References 129
Vita 136

List of Tables

Table 2.1: BLAST programs……30

Table 4.1: DIAT domain statistics for Shewanella genome……85

Table 4.2: DIAT domain statistics for Ecoli genome……86

Table 4.3: DVAT domain statistics for Shewanella genome……92

Table 4.4: DVAT domain statistics for Ecoli genome……92

Table 5.1: BLAST and HMMER suite statistics……100

Table 5.3: Comparison of DIAT on GENWIDEshew and MANGEN files……105

Table 5.4: Computation times of DVAT and DDAT for GENWIDEshew file……105

Table 5.5: Computation times of DIAT, DVAT, and DDAT for GENWIDEecoli file……108

Table 5.6: Statistics for Domains identified for MANGEN file of Shewanella genome using DIAT……111

Table 5.7: Protein statistics in three different ranges……120

Table 5.8: The estimated and computed times for some sample files……122

List of Figures

Figure 1.1: Growth of NCBI database sequences over past few decades…….2

Figure 1.2: Different stages of protein structures, Figure is courtesy of National Human Genome Research (NHGRI), by artist Darryl Leja……6

Figure 1.3: Domains of sensory box protein of Shewanella genome from SMART database……7

Figure 1.4: 3D structure of sensory box protein of Shewanella genome from MODBASE database ……8

Figure 1.5: The regions with arrows represent possible query peptide sequences……10

Figure 1.6: Shared memory architecture……16

Figure 1.7: Distributed memory architecture……..16

Figure 1.8: Cluster architecture……17

Figure 2.1: Distinction between global and local alignments of protein sequences……23

Figure 2.2: Dot plot of a human zinc finger transcription factor……..24

Figure 2.3: The BLOSUM62 matrix……26

Figure 2.4: Scoring diagonal in BLAST algorithm……29

Figure 2.5: Working of PSI-BLAST……..33

Figure 2.6: Multiple sequence alignment of MCP domain against NR database……..35

Figure 2.7: Working of HMMER tool……39

Figure 3.1: BLAST input generator……..54

Figure 3.2: Input query (unknown region) to PSI-BLAST……54

Figure 3.3: The core modules of the DIAT……55

Figure 3.4: Screenshot of section of nr protein database……56

Figure 3.5: Tab-delimited PSI-BLAST output file (PSIOUT) ……..57

Figure 3.6: HMMER input generator……59

Figure 3.7: Screenshot of section of Pfam Database……...61

Figure 3.8:Typical HMMER output file (HMMOUT)……..62

Figure 3.9: The Domain Identification Automated Tool chain flow…….64

Figure 3.10: The Domain Verification Automated Tool chain flow…….66

Figure 3.11: The core modules of the Domain Discovery Automated Tool chain……68

Figure 3.12: The multiple sequence alignment (MSA) input generator……70

Figure 3.13: The Domain Discovery Automated Tool chain flow……72

Figure 4.1: Sequence lengths distribution of Shewanella genome…….76

Figure 4.2: Sequence lengths distribution of Ecoli genome……77

Figure 4.3: Domain Identification Automated Tool chain results flow…….81
Figure 4.4: Resulted DIAT domain distribution of Shewanella genome……82

Figure 4.5: Resulted DIAT domain distribution of Ecoli genome……84

Figure 4.6: Domain Verification Automated Tool chain results flow……..89

Figure 4.6: Resulted DVAT domain distribution of Shewanella genome……..90

Figure 4.7: Resulted DVAT domain distribution of Ecoli genome……..91

Figure 4.8: Automated tool chain flow using Shewanella genome…….95

Figure 4.9: Shewanella domain model with 100% precision with EF-G C-terminal domain from PHYRE search. The red areas indicate alpha helixes, blue areas indicate beta sheets, and gray areas indicate coil regions……..97

Figure 4.10: Ecoli domain model with 0% precision with PDZ domain from PHYRE search. The red areas indicate alpha helixes, blue areas indicate beta sheets, and gray areas indicate coil regions……..98

Figure: 5.1: Comparison plot of PSI-BLAST computation times between Sun Sparc, Intel Xeon, and AMD Opteron……102

Figure: 5.2: Comparison plot of HMMER computation times between Sun Sparc, Intel Xeon, and AMD Opteron……103

Figure 5.3: Domain Identification Automated Tool chain computation time results flow……106

Figure 5.4: Domain Verification Automated Tool chain computation time results flow……107

Figure 5.5: The distribution of protein sequence lengths of 12.8 million protein sequences currently available (Image added with the permission of Luke Ulrich)……113

Figure: 5.6: Comparison plot between hmmpfam computation times and amino acid lengths for 16 different protein sequences of varying lengths from 100 amino acids to 24000 amino acids……117

Figure: 5.7: Three dimensional comparison plot between protein sequence lengths, number of sequences in a file and their respective computation times for hmmpfam jobs……..118

Chapter One

1.1 Introduction

The fields of computational biology and bioinformatics are growing in popularity and demand. Research in bioinformatics and computational biology promises to improve techniques for the prevention, treatment, and cure of diseases [7]. Life sciences research increases the spectrum, demand, and the amount of information generated every year [5]. The best example is the human genome project. There are around 3.2 billion base pairs and 30,000-40,000 protein-coding genes in the human genome alone [54, 55]. There are 401 prokaryotic genomes in Comprehensive Microbial Resource (CMR) database [78] and European Bioinformatics Institute has 53 eukaryotic genomes. This indicates the vast amount of data associated with all genomes that are currently sequenced. The cost and time of sequencing genomes is decreasing with techniques such as shotgun sequencing [56, 57]. This led to sequencing organisms from different phyla. The sequences put in the databases around the world are doubling every six months [40]. The growth of the NCBI (National Center for Biotechnology Information) database sequences is shown in Figure 1.1 [79,80,81].

Many areas in life sciences use the information generated by the genomes for research [4]. There is an information revolution, and the data gathering of

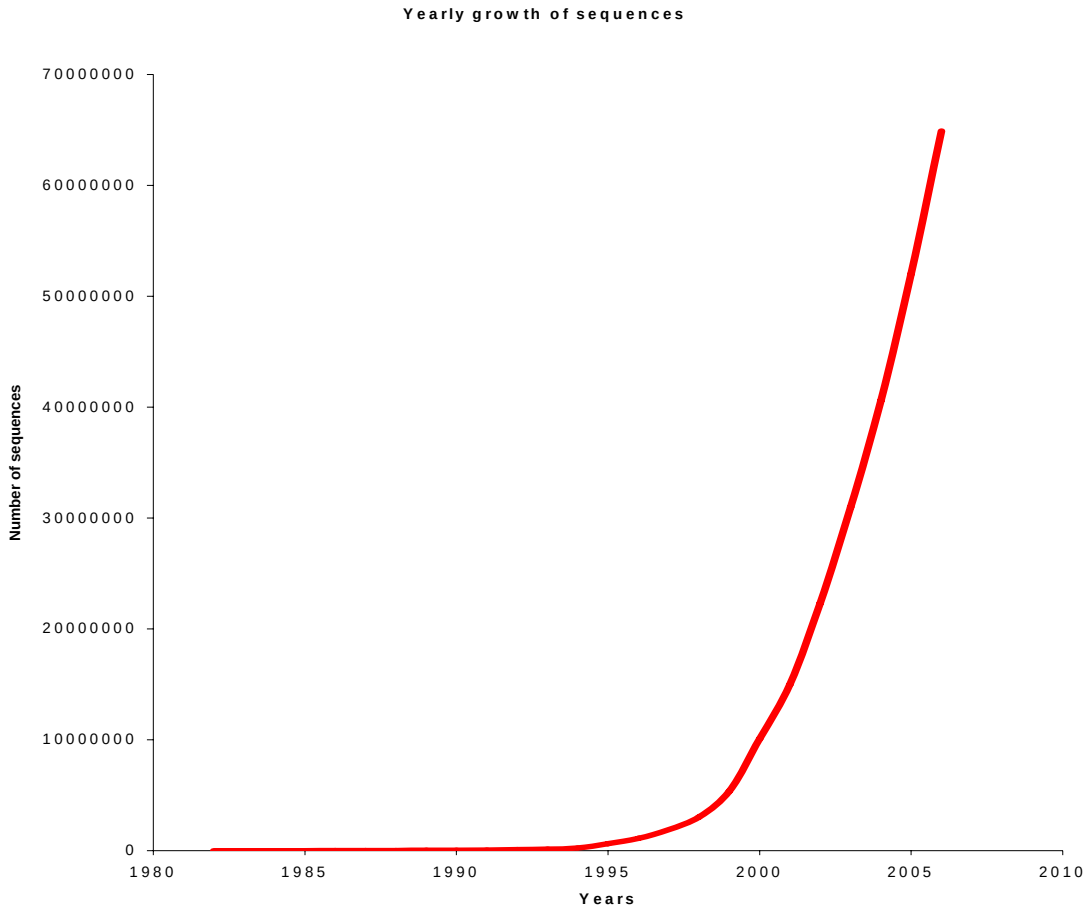


Figure 1.1: Growth of NCBI database sequences [79,80,81] over past few decades.

genomic sequences is increasing at an exponential rate [5, 26], surpassing the data analysis algorithms, architectures, and per-node core memory to handle such a vast amount of data [26, 40]. Some of the major areas of life sciences where an enormous amount of time and money are allocated include sequencing, discovery of new genes, gene ontologies, pathway analysis, regulatory networks of cells, and microarrays. There is a massive amount of data in various databases but there are not enough automated knowledge discovery algorithms or dedicated architectures to mine this data [5]. The major challenge of bioinformatics is to design computer algorithms and architectures to analyze, interpret, and understand all the data within a feasible amount of time.

1.2 Biology Overview

This chapter introduces both engineers and biologists to some basics of biology and engineering to better understand the problem this dissertation addresses. The entire biological and hereditary information of an organism is possessed in the genome. The genomes are made of *DNA (deoxyribonucleic acid),* but for some viruses the genome is made of *RNA (ribonucleic acid)* [82]. The DNA in a nuclear genome is made up of chromosomes. DNA is a double stranded nucleic acid that is made up of *nucleotides* that carry genetic information. These nucleotides are classified into two groups, the *pyrimidines* including cytosine (C), thymine (T) and uracil (U), and the *purines* including adenine (A) and guanine

(G). The DNA is made of AGCT and in RNA uracil (U) takes place of thymine hence RNA is made of AGCU.

The process of transferring genetic information from DNA to RNA is called *transcription*. RNA polymerases are the enzymes which enzymatically transcribe DNA into messenger RNA called *mRNA*. mRNA is used as a template to generate the Amino Acid (AA) sequence of proteins. The process of translating mRNA to proteins is known as *translation*. The study of an entire organism's genome and its genes is known as g*enomics*. On the other hand the study of proteins and their structure and function is known as p*roteomics*. There are 20 amino acids in total and each amino acid is made up of three nucleotides known as a *codon*. In eukaryotes, complex organisms including animals, plants, fungi, and protozoa, the transcription occurs inside the nucleus and translation occurs in cytoplasm outside the nucleus. The ribosome in the cytoplasm of the cell serves as a factory that generates the AA sequences using the mRNA.

This dissertation focuses on proteomics. To better understand the problem, a brief introduction of proteins, their structures and folding is introduced here. There are 20 different kinds of AAs but they all have in common a central carbon atom to which a hydrogen atom, an amino group, and a carboxyl group are attached. Amino acids are distinguished by how the side chains attached to the central carbon atom through its fourth valence. The biological function of the protein can be deduced by the prediction of the three dimensional structure from

the amino acid sequence. A chain of amino acids is called a peptide and peptides are the building blocks of protein structures. Protein structures can be classified into primary, secondary, tertiary, and quaternary shown in Figure 1.2 [84, 85]. The primary structure is a simple amino acid peptide chain. The secondary structures consist of alpha helices and beta sheets that are highly regular, locally defined substructures. The tertiary structures are three-dimensional structures that are spatial arrangements of the secondary structures. The quaternary structure is a complex of two or more polypeptide subunit chains.

Proteins are organized further into smaller units such as motifs and domains. Motifs are common arrangements or combinations of the secondary structural elements. Domains are characterized as semi-independent three-dimensional functional and evolutionary units of proteins [13,14,15]. Protein folding is the process by which a protein acquires its three dimensional structure to achieve the biologically active native state. Protein folding is a major intellectual challenge in life sciences and biology that is yet to be solved [83]. The study of protein folding is very important as misfolding can lead to various diseases. As there are 20 different amino acids that can be combined in many possible combinations, protein folding prediction remains a huge problem. Hence the protein structures are determined experimentally using various techniques such as x-ray crystallography, electron crystallography, or nuclear magnetic resonance techniques.

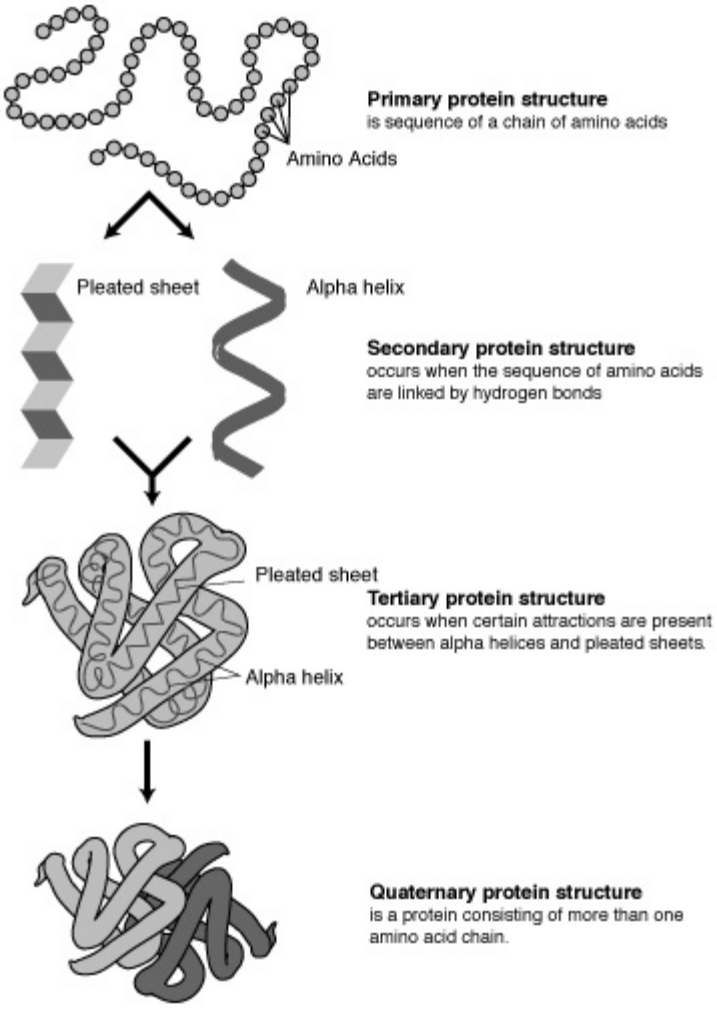


Figure 1.2: Different stages of protein structures [84, 85], Figure is courtesy of National Human Genome Research (NHGRI), by artist Darryl Leja

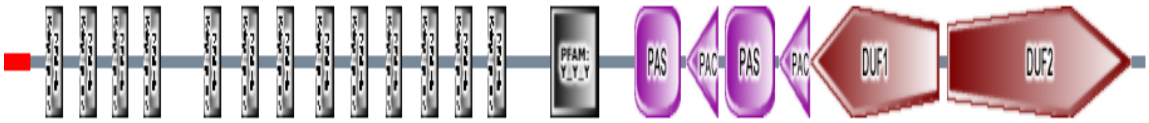


Figure 1.3: Domains of sensory box protein of Shewanella genome from SMART database for potential domains

Protein domain discovery is another very important part of life sciences used for protein classification, predicting protein structure, function, evolution, and modeling [13,14,15]. Figure 1.3 shows the domain information of a sensory box protein of the *Shewanella oneidensis* genome (referred to as Shewanella in the rest of the dissertation). Shewanella belongs to the bacteria phylum and the 3D structure of a sensory box protein is shown in Figure 1.4. Protein domains have limits on their sizes, ranging from around 40 amino acids (AAs) to around 700 AAs, averaging approximately 100 AAs, although the sizes vary [16,17,18].

Different labs around the world are exploring various types of genetic information; one popular field is microarray analysis and another is computational genomics. Microarray analysis is expected to produce a peta-byte of data per year [40]. Microarrays can house the genes of an entire genome on a single glass slide. The microarray technology allows researchers to follow the expression of an organism's entire complement of genes simultaneously in a single experiment [74,75,76,77]. There are different types of microarrays such as gene arrays, protein arrays, transcription factor arrays, and also DNA microarrays, thus

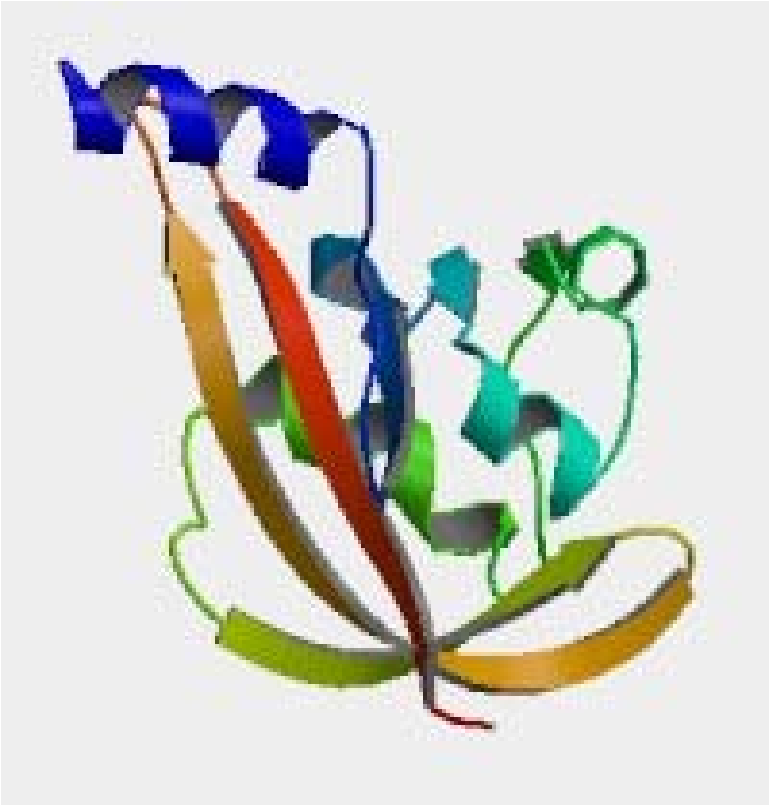


Figure 1.4: 3D structure of sensory box protein of Shewanella genome from MODBASE database [61].

populating the databases with a plethora of data [5]. Computational genomics is used to study the evolution, diversity, and molecular mechanisms of functions such as signal transduction.

The microbiology research group with which we are collaborating is interested in solving the problems related to signal transduction in prokaryotes. The group is interested in prokaryotic organisms, as they are simpler than eukaryotes but sophisticated enough to adjust to environmental changes using detectors and transmitters. One of the challenges in genomics is to derive relevant information for complete sequenced genomes and this dissertation addresses one such problem. This dissertation addresses the problem of protein domain discovery on a genome-wide scale using various computing architectures.

1.3 Problem Overview and Motivation

1.3.1 Problem

There is a vast amount of knowledge that is yet to be discovered in the field of proteomics to better understand evolution, structure and function of the proteins. Domains are the key elements of the proteins that aid in understanding structures, functions, and evolution of proteins. Currently there are around 1200 bacterial and archaeal genomes in the MiST (Microbial Signal Transduction)

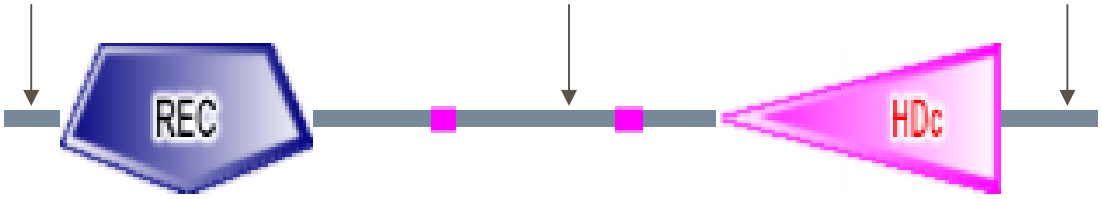


Figure 1.5: The regions with arrows represent possible query unknown sequences for potential domains

database [51]. There are roughly 5000-proteins in each genome; these proteins have one or more known domains. The domains are identified using both the Pfam (protein families) [64] and SMART (simple modular architecture research tool) [32] databases by the HMMER tool. There are peptide regions in these proteins that are greater than 80 AAs and for which no domains are identified by the HMMER tools, known as unknown regions. These unknown regions lie between two known domains, or between the start of the protein sequence and the domain, or after the domain to the end of protein sequence as shown in Figure 1.5. These unknown regions have a potential of being a new domain or the result of a poor domain model and hence not identified by HMMER tool.

Discovery of new domains is a tedious and manual procedure. For example the FIST domain [87] was discovered after months of research including hundreds of profile searches, multiple sequence alignments, structure prediction, and domain architecture analysis by one student. At this rate, it will take years of effort to

model all the unknown domain regions of a single genome. Hence the need of bioinformatics tools and computer architectures arises to increase the rate of domain discovery.

1.3.2 Algorithm

One practical way to find the domains for these unknown regions is to perform a PSI-BLAST search on the unknown region to find the relatives or similar sequences for this region. On all the matching regions from the PSI-BLAST search, a domain identification check is performed using the HMMER tool against the Pfam and SMART databases. PSI-BLAST and HMMER are robust and sensitive searching tools that use principles of full probabilistic modeling to build models from multiple sequence alignments [36]. PSI-BLAST is sensitive and discovers new and interesting protein sequence alignments because of its iterative functionality. The HMMER tool is a widely used and important tool for protein domain identification. PSI-BLAST and HMMER are run on the query unknown region simultaneously for protein domain identification.

Further analysis is performed on all the query sequences for which no domains are identified in the above process. The MUSCLE tool is used to build multiple sequence alignments and the HMMER tool is used to build the HMMs (Hidden Markov Models). These newly built HMMs are searched against the non-

redundant (nr) database, until all the protein regions that match this HMM model are retrieved. Then a final check is performed for known domain models in the protein regions resulting from the previous search. If no known domains are identified, this implies a potential new domain model is built. Now structure predictions and domain architecture analysis are performed to define this new domain model. All the above processes PSI-BLAST, HMMER, and MUSCLE are performed manually right now, which is tedious and time consuming.

1.3.3 Challenge

The problem size is enormous, for 1200 genomes in the MiST database alone and assuming there are around 2000 peptide regions per genome with no domain information, one would have to run 1200*2000 PSI-BLAST searches and perform HMMER searches on the protein regions that resulted from the PSI-BLAST searches to identify domains. Apart from that, for all the unknown regions for which no domains are found, multiple sequence alignments and HMM models need to be built and searched against the nr database. The same approach can be extended to the entire collection of known proteins available in different databases around the world to construct a peptide domain database for all the peptides along with discovering new and interesting domains.

This dissertation deals with combining the PSI-BLAST, HMMER, and MUSCLE tools for protein domain identification and discovery. New bioinformatics automated tool chains are proposed for domain identification and discovery to enable millions of searches. The embarrassingly parallel nature of this automated tool chain is exploited. This led to using cluster-computing techniques to increase the rate of domain discovery. A brief introduction about computers for biologists is introduced in next section to better understand the approach used to solve the problem later in this dissertation.

1.4 Hardware Architectures Overview

The sequences in the databases are growing at an exponential rate [26], and doubling their size every six months [40]. According to Moore's law [27] the number of transistors on a chip double every 18 months. Hence the growth of single processor speeds (hardware growth) is not able to keep up with the rate of sequence growth. The latest trend in processors is the multicore technology, where one or more cores are fabricated on the same chip. The success of the dual-core technology led to the development of quad-core or more number of cores on a single chip [89]. The magnitude of the biological data is so abundant that a single processor cannot solve it. This led to the use of clusters of computers and supercomputers for sequence analysis along with hardware accelerators such as Field Programmable Gate Arrays (FPGAs) and Application

Specific Integrated Circuits (ASICs). Many biological problems are embarrassingly parallel in nature, thus the use of a parallel cluster of workstations is an effective solution [90]. One good example is the Folding@Home project dedicated to understand protein folding to cure diseases by using the processor cycles of the participants' workstations around the world. Addressing problems on a genome-wide scale is a grand challenge that can be solved using High Performance Computing (HPC). Two such challenges, understanding evolution and discovering protein structure and functions can be solved only through use of high-performance computing [91, 92].

Traditional computers with a single CPU exploit serial computation. Serial computing involves executing one instruction after the other in order, to complete the problem. In the modern world of computers the size of the applications is increasing at a much higher rate than the resources can accommodate individually and the speed with which the applications should be executed is becoming higher. These fast growing requirements of the life sciences, engineering, database, commercial, and business applications and lower time to market led to the development of many techniques of computing. One option is to update all the systems available, which increases the cost. The second is to use the already existing systems intelligently, where the need for parallel computing arises. There is an immense necessity for parallel computing in the latest world of computing technology and it is becoming the dominant technique in achieving high performance. As the demand of the processing power

increased due to the advent of challenging sized problems such as weather and climate, chemical or nuclear reactions, biology etc, necessity to reduce the computation time arises. Parallel computing has emerged to compete with the existing supercomputers [90, 92].

Parallel computing in a simple sense is to simultaneously use multiple processors in a computer or multiple computers connected on a network or both, to solve a computational problem in lesser time. There are two types of parallelism: data parallelism and functional parallelism. Data parallelism is concurrently running the same operations on different sets of data. In contrast, functional parallelism consists of concurrently executing different operations on a single stream of data [94, 95]. Some basic terms involved with parallel computing are efficiency, speedups, bandwidth, latency, and task or job.

Memory architectures play a major role in parallel computers. Two major approaches are shared memory architectures where all the processors use one global memory and distributed memory architectures where each processor has its own local memory as shown in Figure 1.6 and 1.7. The largest and fastest computers now use combination of both shared and distributed memory architectures. For example the cluster used for running automated tools is built with Intel Xeon dual core processor with 4GB of RAM, as in Figure 1.8. There are several parallel programming models such as shared memory, threading, data parallel, and message passing. This dissertation also explores the computation

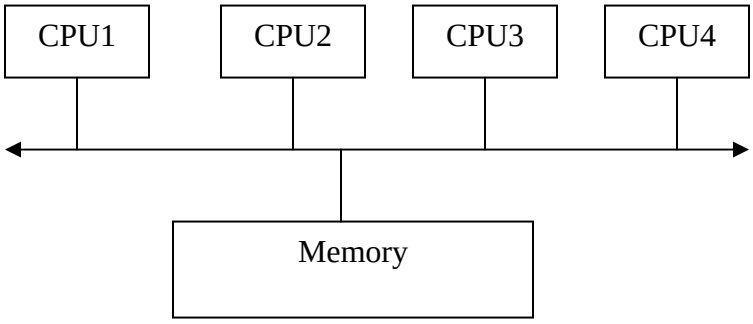


Figure 1.6: Shared memory architecture

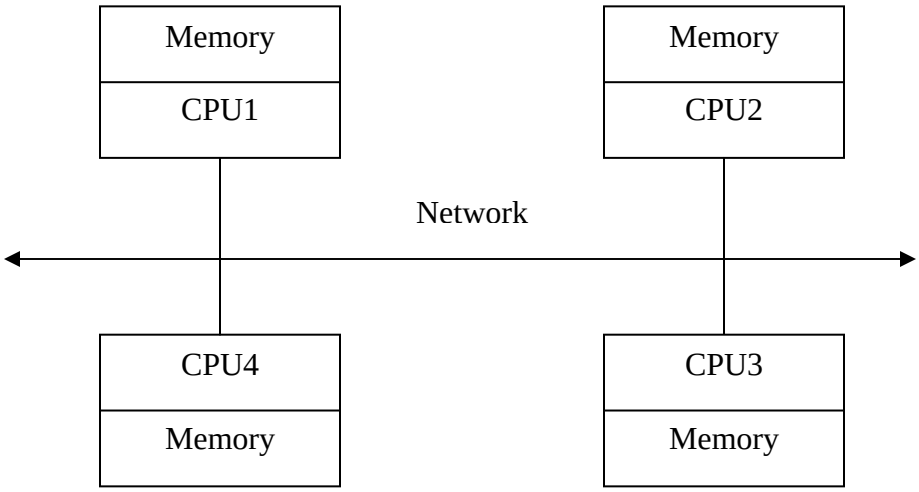


Figure 1.7: Distributed memory architecture

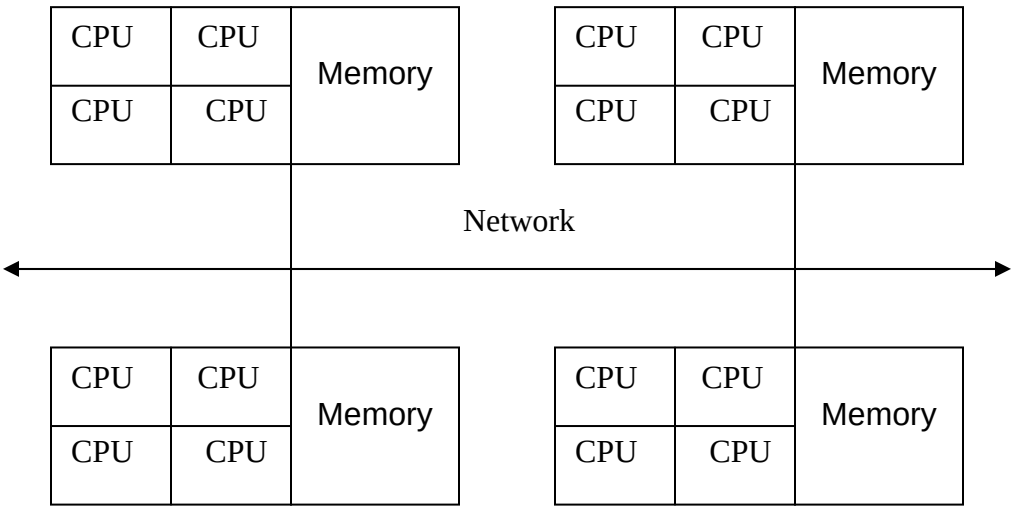


Figure 1.8: Cluster architecture

time taken by the tools used for sequence analysis on single core versus multiple core processors along with various memory architectures for performance evaluation that are discussed in later chapters.

Traditional computing can be further divided into general purpose computing and application-specific computing. Microprocessors are used for the general purpose computing. Microprocessors perform a wide range of applications and they are flexible. On the contrary they are slow for few applications that require huge data processing. This is because of the architecture of the microprocessor is fixed. Which means they include the hardware that can perform a limited and predefined set of instructions present in the memory and execute them. This results in high execution overhead in each operation, thus making it slow. In the

microprocessors, the software programs determine the computation, as the hardware is fixed.

On the other hand, for application specific computing, ASICs or FPGAs are used, which are used to perform the operations in hardware. The ASICs are designed specifically to perform the operations in hardware. The ASICs are fabricated to a particular digital design to perform an application and the design cannot be altered. So the ASICs are very fast and efficient in performing specific operations for which they are designed but with limited flexibility. The other disadvantages of ASICs are high design and fabrication time, along with high cost. However, they are used in the application in which the speed is an important design consideration and economies of scale.

The flexibility of the microprocessor and the speed of the ASICs can be achieved with Reconfigurable Computing (RC) architectures. The main components of the RC systems currently used are FPGAs. The advancements in the design of the FPGAs lead to a drastic improvement in the RC systems. The FPGA basically consists of the programmable logic blocks and programmable interconnects. The currents FPGAs have static random access memory (SRAM) cells for configurations, which improves the flexibility in the design. The SRAM FPGAs are easily re-programmable when compared to one-time programmable devices like ASICs or antifuse FPGAs. Thus, the bug fixes or upgrades are easily possible, hence providing an ideal prototyping medium.

The automated tool chain designed to solve the domain-modeling problem can deal with multiple sequences at a time, allotting one sequence to a node or processor. The embarrassingly parallel nature of this biology problem is exploited to discover new domains for thousands of proteins found every year in less time. When addressing a problem on genome-wide scale there are thousands of sequences to manipulate. Thus allotting jobs to available nodes on a cluster can become very important part of the research, especially when allotting jobs on a very large scale on a supercomputer. The jobs are divided efficiently so that the load on the computers are balanced so that no one computer will become a bottleneck for analyzing data that is described in more detail in later chapters. The next section presents the scope of this dissertation.

1.5 Scope of Dissertation

Chapter one introduces the reader to basic concepts of biology and computers along with the problem overview and challenge. Chapter two discussed the background, literature review, and some related work. Chapter two also describes the algorithms of the tools used to build the automated tool chain to solve the domain-modeling problem. The automated tool chain designs for domain identification, verification, and discovery are described in chapter three along with the various databases designed to reduce the computation time.

Chapter four illustrates the biological results obtained for Shewanella and *Escherichia coli* (referred to as E.coli from here on) genomes. Chapter five describes some performance metrics, threading issues, job mapping algorithms, and computation times. Finally chapter six concludes with contributions, conclusions, and proposed future work to further enhance the automated tool chains that was beyond the scope of this dissertation.

Chapter Two

Literature Review

This chapter describes the basic concepts of biology required for protein domain modeling, such as sequence alignments, profile searches, multiple alignments, and secondary structure predictions. This chapter explains the essential steps involved in understanding the biological problem of interest. The key concepts of algorithmic and architectural advancements of the bioinformatics tools such as BLAST and HMMER used to design the automated tool chains for protein domain modeling are explored further.

2.1 Biological Background

This dissertation deals with important areas of proteomics such as sequence alignments, multiple sequence alignments, HMMs, and domain identification that are the key elements for protein domain modeling along with secondary protein structure prediction for verification. The rest of the section explains the key elements, algorithms, and tools used to build the automated tool chain for protein domain modeling.

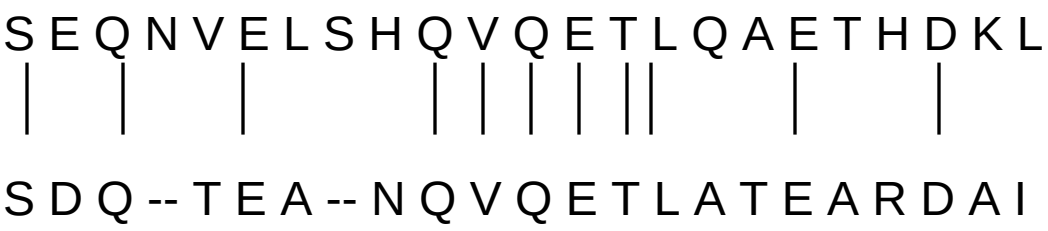
2.1.1 Sequence Alignments

Sequence similarities may be the consequence of structural, functional, and evolutionary relationships between the sequences. From the alignment of two sequences one can infer the evolutionary relationship, functional domains shared between proteins, and transcription-factor binding sites for DNA sequences. The functional and evolutionary diversity can be recognized from distant sequence relationships.

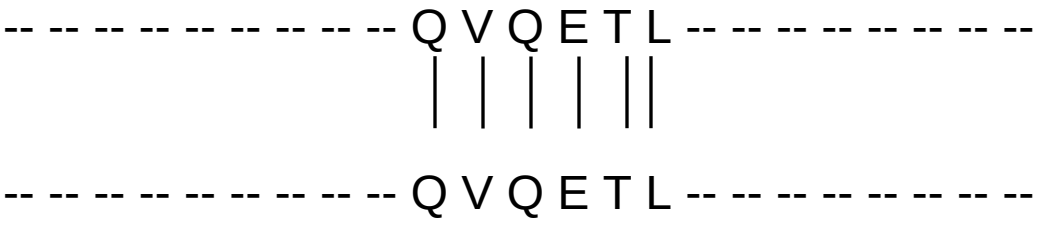
There are two types of sequence alignments,

- **a.** Pair-wise sequence alignment: two DNA or protein sequences are compared by searching for series of individual characters or character patterns that are common.
- **b.** Multiple sequence alignment: a nucleotide or protein sequence is compared with two or more sequences to identify regions of similarity.

The pair-wise sequence alignments are further classified into global alignments and local alignments. In global alignment, the full length of the sequence is aligned using all sequence characters. On the other hand, local alignment is concentrated on the stretches of sequences with the highest density of matches. Figure 2.1 illustrates global and local alignments. Three principle methods of pair-wise sequence alignments used in common are dot matrix analysis, dynamic programming, and word or *k*-tuple methods.



Global alignment



Local Alignment

Figure 2.1: Distinction between global and local alignments of protein sequences

The dot matrix method is considered to be the first choice for pairwise alignments unless two sequences are known to be similar, because for its graphical display as in Figure 2.2. This method is time consuming to analyze large sequences, but good for revealing the presence of insertions, deletions, and repeats. The dot matrix plot is constructed using two sequences that are to be matched. The top most row and the left most column of the matrix are populated using the two sequences. A dot is placed at a point where the characters in the appropriate column match. Very closely related sequences will appear as a single line along the matrix's diagonal in the dot matrix plot as shown in Figure 2.2.

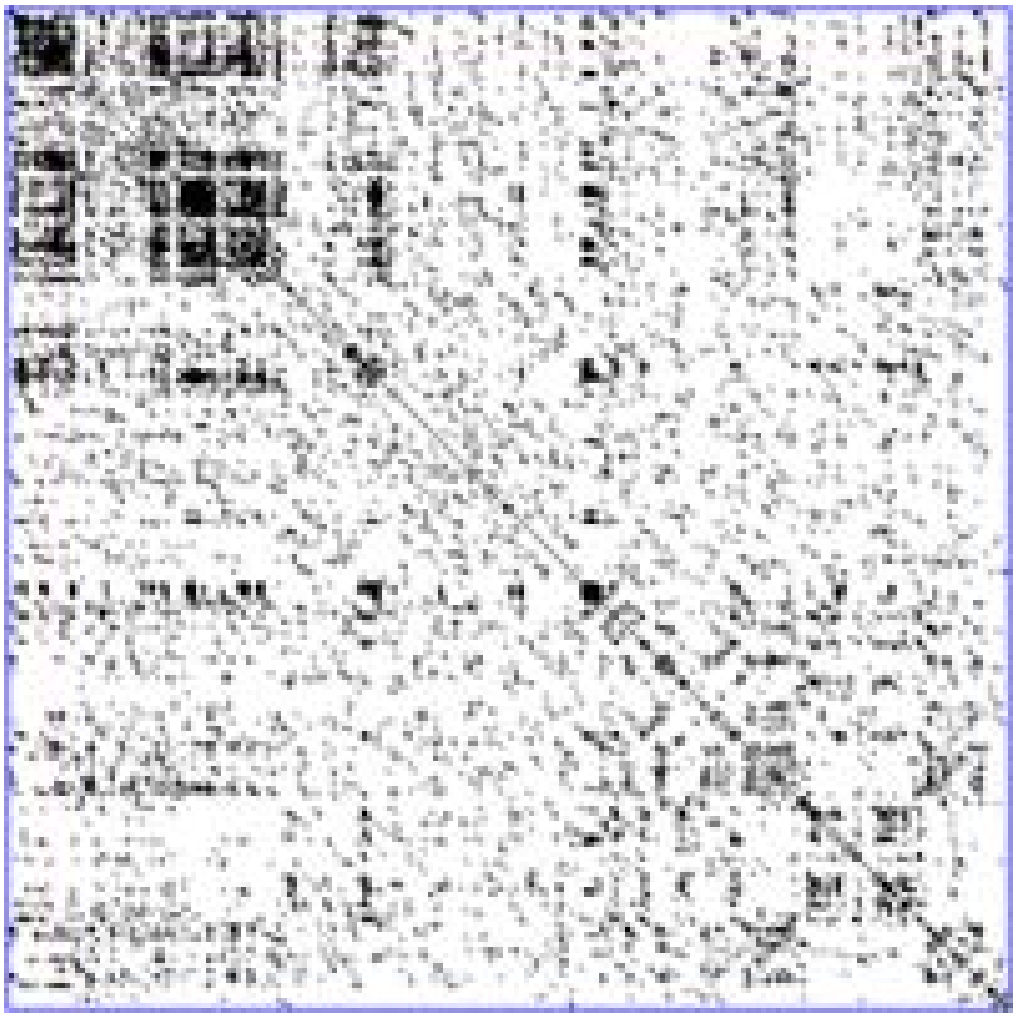


Figure 2.2: Dot plot of a human zinc finger transcription factor [120]

Needleman and Wunch [11] were first to use dynamic programming algorithm for global alignment of protein sequences. Smith and Waterman [10] were first to use dynamic programming algorithm for local alignment [7]. Matching all possible pairs of characters between the sequences by using a scoring scheme for matches, mismatches, and gaps generate the alignment. This procedure generates a matrix of numbers and the highest set of sequential scores in the matrix defines the optimal alignment. This matrix looks like a normal matrix of numbers but aligning two sequences one along the vertical axis and other along the horizontal axis as shown in Figure 2.3. For DNA and RNA alignments, a positive match score, a negative mismatch score, and a negative gap penalty are used for building the matrix. For proteins, a substitution matrix such as the percent accepted mutation matrix 250 (PAM250) [59] or the blosum substitution matrix 62 (BLOSUM62) [58], are used to score matches and mismatches to build the matrix. The dynamic programming algorithm generates optimal alignments at the cost of more time due to the large number of computational steps.

The word or *k*-tuple method is heuristic [7]. First, a search is performed to identify short stretches of nonoverlapping subsequences known as word or k-tuple between sequences. These words are used to join into alignment using the dynamic programming method. This method is not guaranteed to find an optimal solution but is significantly faster, efficient, and statistically reliable to provide the best scoring alignment possible. The fast nature of this method has made it

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

Figure 2.3: The BLOSUM62 matrix [58].

suitable to search an entire database of sequences for similarities. The following section discusses various tools available for finding sequence similarities.

2.2 Pair-Wise Sequence Alignment Algorithms and Tools

Sequence alignment algorithms are one of the most widely used algorithms in bioinformatics for finding functional, structural, or evolutionary relationships between sequences [21]. Some popular sequence alignment algorithms include dot matrix [9], Smith Waterman [10], Needleman and Wunch [11], FASTA [12], and BLAST (Basic Local Alignment Search Tool) [1,2]. These sequence alignment algorithms use optimized methods such as dynamic programming,

heuristic, and probabilistic as a backbone to search huge genomic databases [7]. The sequence alignment tools such as Smith Waterman, Needleman and Wunch use dynamic programming, BLAST use heuristic methods, and HMMER use probabilistic methods. The most popular and widely used sequence-searching algorithm is the BLAST algorithm because of its speed, efficiency, and sensitivity [2,7]. This dissertation uses the BLAST suite for generating the protein sequence similarities that are described in the following section.

2.2.1 BLAST Algorithm

The BLAST algorithm is a heuristic method used for sequence similarity search. BLAST [1] is faster than dynamic programming methods and FASTA [70], while at the same time is also considered to be as sensitive [7]. The BLAST tool is publicly available through the NCBI (National Center for Biotechnology Information) website and also available through a number of other websites, thus making it more popular than other sequence alignment search algorithms [1,2]. The BLAST algorithm first generates the common words or k-tuples in the query sequence and each database sequence. The length of each word is 3 amino acids (e.g. LEA) for proteins and 11 nucleotides (e.g. ATTCGGATCGA) for DNA sequences. The alignment score is calculated using substitution matrices such as Blosum62 [58] or PAM250 [59], for the match between the words of the query sequence and database sequences [7]. This score should be high enough for

significant matches but not too high to miss short but significant patterns [1]. The BLAST algorithm can be used for both gapped and ungapped sequence alignment searches. The newer gapped alignment searches are more popular, as it runs at approximately three times the speed of the original BLAST [2]. Once the significant words are detected, the query sequence is expanded. A cut-off score is used as a threshold to pick the most significant matches. Then the alignments are extended on either direction of the matching words along the sequence as long as the score increases; the extension process is stopped once the score is decreasing thus forming a high scoring segment pair (HSP). In the recent gapped BLAST [2], the threshold was decreased and sequence alignment was only extended if two significant words are lying on the same diagonal (see Figure 2.4) and within a specified distance, thus increasing efficiency of BLAST. Then based on the statistical significance and expected value 'E' ("the E value is the chance that a score as high as the one observed between two sequences will be found by chance in a search of a database of size D" [7]) the final alignments matches are outputted to the results file.

2.2.2 BLAST Suite

The BLAST suite provided by NCBI has different types of BLAST searching programs for different types of protein and nucleotide databases as shown in Table 2.1.

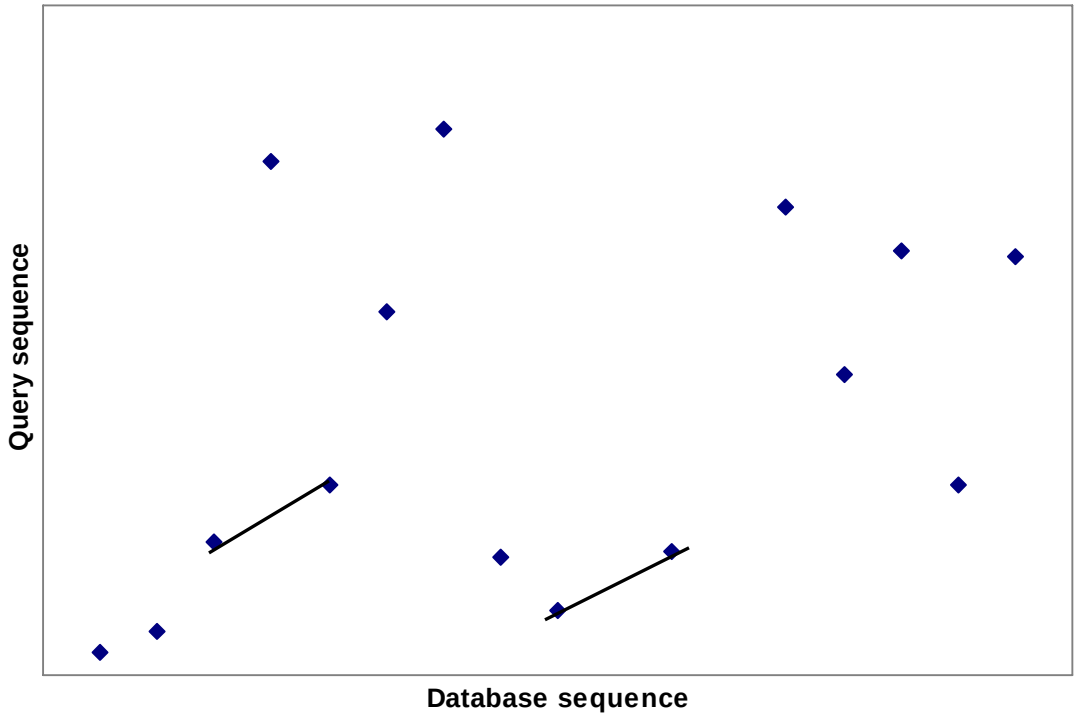


Figure 2.4: Scoring diagonal in BLAST algorithm

Table 2.1: BLAST programs.

Program	Query sequence	Database
BLASTP	Protein	Protein
BLASTN	Nucleotide	Nucleotide
BLASTX	Translated nucleotide	Protein
BLASTN	Protein	Translated nucleotide
TBLASTX	Translated nucleotide	Translated nucleotide

The NCBI databases consist of protein sequences and nucleotide sequences for different organism and genomes from various public and independent resources in the world. Some popular sequence databases other than NCBI are SwissProt [71], European Molecular Biology Laboratory (EMBL) [72], and the DNA Databank of Japan (DDBJ) [73]. The NCBI databases also consist of conserved domain databases (CDD). The exponential growth of sequences in the NCBI database is shown in Figure 1.1.

DNA sequences have only four bases, whereas protein sequences consists of 20 amino acids (AAs), thus resulting in a larger variety of sequence characters in proteins. This increased complexity makes it easier to detect patterns of sequence similarity between protein sequences when compared to DNA sequences [7]. Thus protein sequence database searches yield more significant matches when compared to DNA sequence databases for a specific protein sequence [8].

2.2.3 PSI-BLAST

Protein search significance shows the importance of PSI-BLAST (Position Specific Iterative BLAST), which uses iterative BLASTP. Iterative BLASTP searches are more sensitive to locate conserved domains in query protein sequences, which is the main focus of this dissertation.

The first iteration of PSI-BLAST is BLASTP with the standard substitution matrix, a matrix containing values proportional to the probability that one amino acid is replaced by another amino acid for all pairs of amino acids. PSI-BLAST uses the gapped BLASTP program for searching the query protein sequence against the protein database. Once proteins similar to the query sequence (known as relatives) are found, PSI-BLAST constructs a profile and multiple alignments based on these relatives. This profile is then compared to the protein database to seek local alignments using the BLASTP program. In the second iteration once the local alignments are constructed, PSI-BLAST estimates their statistical significance to find new relatives. Now a new profile is generated and PSI-BLAST iterates using this new profile. The process is repeated for a given number of iterations or until no new relatives or protein sequence matches are found thus reaching convergence [2, 60]. In this research we used four iterations as they are sufficient for sensitive homology searching, and any more iterations

may lead to profile wander [29, 52]. Figure 2.5 illustrates PSI-BLAST.

2.3 Multiple Sequence Alignment Algorithms

Sequences of different organism are often related. Based on the evolutionary process genes are conserved across widely divergent species. These genes sometimes perform the same functions and sometime mutate to perform different functions through the application of natural selection [7]. Hence multiple sequence alignments play a major role in assessing the sequence conservation of structural and functional properties among the family of sequences. Multiple alignments of proteins are used for applications such as phylogenetic tree estimation, secondary structure prediction, and critical residue identification [103]. In this dissertation the multiple sequence alignments of protein sequences are used to build HMMs and for secondary structure prediction. Multiple sequence alignments are more difficult than pairwise alignments, thus MSAs require sophisticated methodologies such as heuristic methods. Some popular MSA tools are CLUSTALW [99], T-COFFEE [100], MAFFT [101], PRALINE [102], MUSCLE [103], and DIALIGN-T [104]. This dissertation uses CLUSTALW and MUSCLE for building MSAs as described in the following section. The

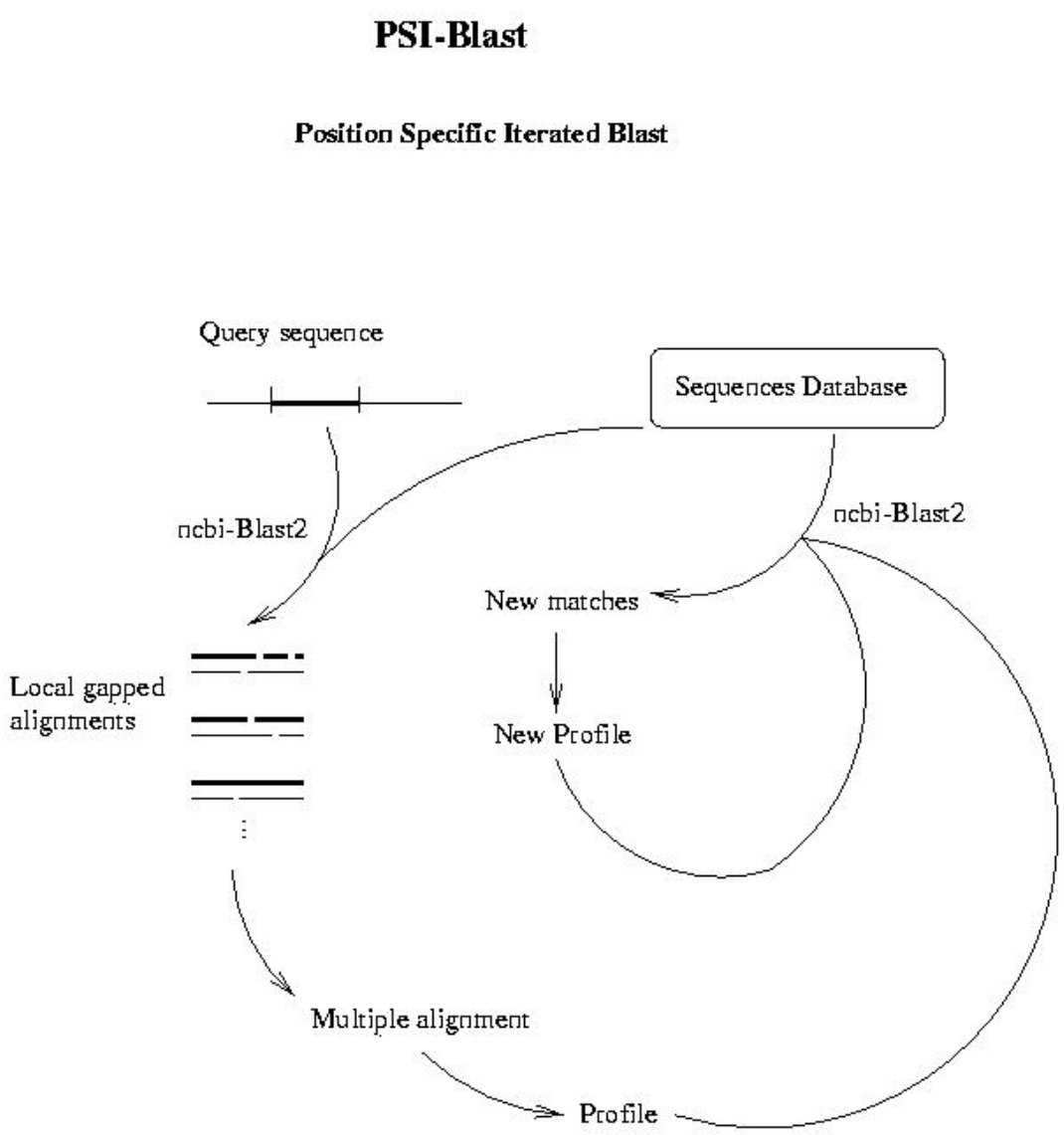


Figure 2.5: Working of PSI-BLAST [2, 98].

multiple sequence alignment of the MCP domain against the NR database is shown in Figure 2.6.

2.3.1 CLUSTALW and MUSCLE

CLUSTALW is a popular multiple sequence alignment program available since 1988, where W represents the weights allocated to the sequences. CLUSTALW first performs pair-wise alignments of the sequences. Then the phylogenetic tree is built based on the alignment scores of genetic distance between the sequences. Finally dynamic programming is used to align the sequences sequentially [99]. This implies most closely related sequences are aligned first and other sequences are added to this alignment.

MUSCLE has three stages: draft progressive, improved progressive, and refinement. At each stage a MSA is generated and improved in the succeeding stage. In stage one, a draft MSA is generated first from the similarity measure using the k-mer counting or global alignments. The triangular distance matrix is built from the pairwise similarities and a tree is constructed from the matrix. An alignment is built by following the tree to the root. In stage two, a similarity measure is computed using the fractional identity computed in the previous multiple alignments. A tree is constructed using a Kimura distance matrix (defined below) and this tree is compared with the previous one. Stage two

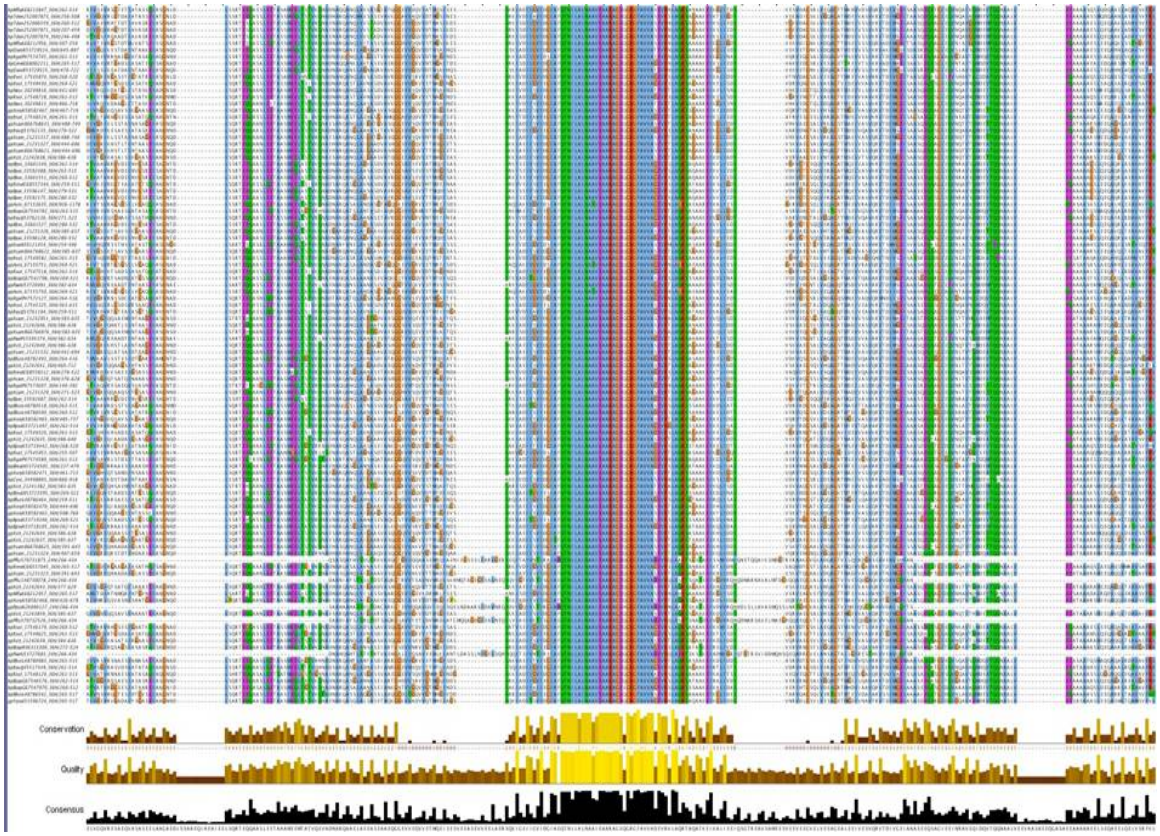


Figure 2.6. Multiple sequence alignment of MCP domain against NR database.

iterates until the tree converges when a new progressive alignment is built. In stage three, the multiple alignment is further refined and the process iterates a user specified number of times unless it converges.

Kimura distance: This is a rough-and-ready distance formula for approximating PAM distance by simply measuring the fraction of amino acids, p, that differs between two sequences and computing the distance as [115,116]

$$D = \log_e (1 - p - 0.2 p^2)$$

MUSCLE is used because of its speed and accuracy; MUSCLE is as accurate as CLUSTALW [99] and takes two to three orders of magnitude less time than CLUSTALW [103]. On the other hand CLUSTALW is also used in this dissertation as some of the secondary structure prediction tools take the alignment only from the CLUSTALW program.

2.4 Profile Hidden Markov Models and Protein Domain Identification

A multiple sequence alignment of homologous sequences is generated using a Hidden Markov Model (HMM) considering different possible combinations of matches, mismatches, and gaps. A profile HMM represents a multiple sequence alignment profile. The profile HMMs are used for gene finding, sequence

composition and pattern analysis, phylogenetic analysis, and protein secondary structure prediction [36]. Sequence Alignment and Modeling Software System (SAM) [37, 38] and HMMER are commonly used HMM tools. HMMER operations rely on accurate construction of the profile HMMs. These HMMs are applied to protein sequence databases for homology determinations used for extending the protein families that are used for finding functional annotations of query sequence. HMMER functions are based upon a profile HMM architecture which is constructed using a plan-7 model [39]. The plan-7 architecture is constructed using the Viterbi algorithm [37, 39]. The HMMER tool package is used to search the protein sequences against the protein domain databases of HMM models and identifies the protein domains in the query protein sequence [36].

2.4.1 HMMER Suite

The HMMER package is widely used for the detection of protein sequence homology, functional annotation, and protein family classification. It uses profile Hidden Markov Model (HMM) methods for sensitive database searches. Multiple sequence alignments are used as search queries to build statistical models for database searches. The HMMER package has different programs for use [28].

- Hmmbuild: builds profile HMMs using multiple sequence alignments
- Hmmcalibrate: calculates accurate expectation values for sensitive database searches

- Hmmsearch: searches for new homologies using the profile HMMs
- Hmmpfam: identifies protein domains
- Hmmalign: aligns multiple sequences to an existing model
- Hmmindex: indexes the HMM database
- Hmmfetch: extracts a model from HMM database
- Hmmconvert: converts file formats
- Hmmemit: emits sequences from the HMM database

Hmmpfam is used for detecting known domains in a query sequence by searching against the library of profile HMMs such as the Pfam and SMART databases. The input to the HMMER tool is the file with the query sequence or a batch file with multiple sequences, which is searched against the database of HMMs. Query sequences are searched one at a time and each search is independent of the other. Thus, HMMER searches can be easily made concurrent by exploiting embarrassing parallelism similar to BLAST searches. Figure 2.7 shows the working of the HMMER programs [105].

2.4.2 Domain Identification Tools

Different domain resources such as Interpro [62], PROSITE [63], Pfam [64], PRINTS [65], ProDom [66], SMART [34], and CDD [67] are available [58]. Many

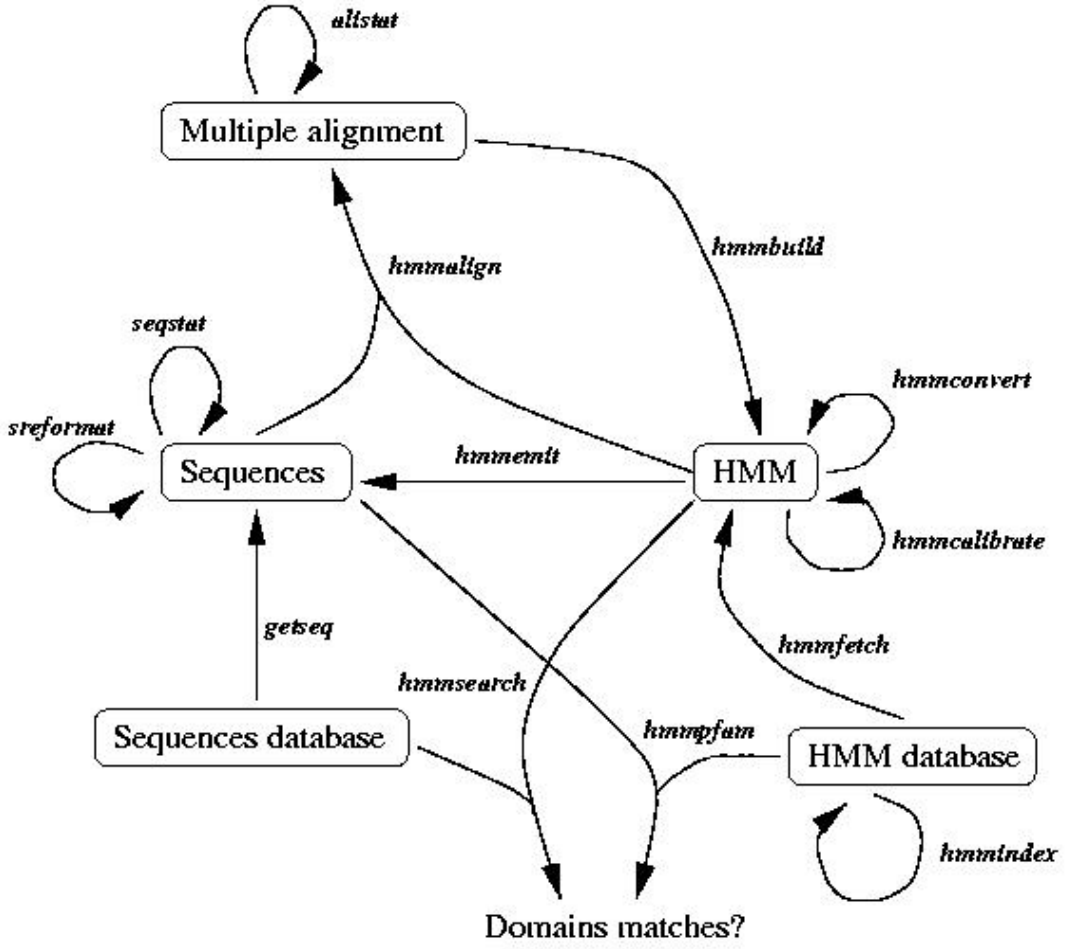


Figure 2.7: Working of HMMER tool [105].

techniques and algorithms are proposed for protein domain identification and discovery; this dissertation surveys them. DOMAINATION, a web-based tool, recognizes domain insertions and permutations [29]. DOMAINATION uses PSI-BLAST along with methods to cut the query sequence into domains. Once the query sequence is cut into domains, PSI-BLAST is run on each domain to generate a Multiple Sequence Alignment (MSA) and the results from PSI-BLAST are then used for further database searches. The process is repeated until no more new sequences are found by PSI-BLAST or domain cutting finishes.

MyHits is an interactive web server with resources for protein annotations and domain identification [30]. There are two different types of MyHits users. One is a guest user who can access searches only for publicly available databases and another is a requested user who has access to both private and public databases. MyHits includes standard bioinformatics tools for use along with protein motif databases. These databases contain pre-computed lists of matches between the sequences and motif databases [30].

THOR is another web-based tool for domain discovery [31]. THOR compares the HSPs (High Scoring Pairs) generated by all significant hmmsearch searches of the HMMER tool. New alignments and HMMs are built using hmmalign, hmmbuild, and hmmcalibrate. This iterative procedure is repeated until no new entries are added to the alignment to generate a final alignment, which is used for domain discovery [31].

SMART (Simple Modular Architecture Research Tool) [32] is another web-based tool for protein domain identification and analysis of domain architectures with an emphasis on eukaryotic mobile and signal transduction domains. Later SMART included extracellular GPS and PSI domains, intracellular signaling domains, and splicing factor domains with all the members of a domain family having complete taxonomic information. SMART uses HMMER2 to search their HMMs [32, 33].

Lachlan et al [35] used the HMMER tool to search the Pfam database. They included the knowledge of the taxonomic distribution of protein domains for searching the Pfam database to enhance protein domain recognition, which was validated using PSI-BLAST. They found 4447 new instances of Pfam domains in the SP-TREMBL database by including the taxonomic distribution [34].

DOMPRO employs machine learning in the form of recursive neural networks for domain identification [35]. The neural networks are a combination of profile derived from evolutionary information using PSI-BLAST predicted secondary structures and predicted relative solvent accessibility in a 1D-recursive neural network.

These are commonly used tools and algorithms for protein domain identification. Various algorithmic and architectural improvements for the BLAST and HMMER tools are described in section 2.6.

2.5 Secondary Protein Structure Predictions

Secondary structure predictions help in deriving protein structures and functions. The secondary structure prediction applications assist in classifying proteins, separating domains, and identifying functional motifs [106]. The secondary structure predictions can be particularly helpful in determining tertiary structures via fold recognition methods [107, 108]. New concepts and approaches are proposed for secondary structure predictions and improvements. Some of the popular approaches that claimed to have higher accuracy are the combination of neural networks and the position-specific scoring matrix generated from PSI-BLAST, a support vector machines approach, and a simple statistical model approach [106].

Various tools used for secondary structure predictions are PSIPRED [109], Sspro [110], PROF [111], Jpred2 [112], PHD [113], and SVMpsi [107]. According to Burkhard Rost [108] 88% is the limit of prediction accuracy; according to his review PROF and PSIPRED are the two tools best attaining 77% and 76.6% of accuracy. The SVMpsi claims to have attained 78.5% accuracy, the highest reported.

The prediction method in PSIPRED is divided into three stages. In the first stage PSI-BLAST is used to build profiles using the custom sequence database instead of the nr database. In the second stage a neural network architecture is used to

build the initial secondary structures. Finally the predicted structures are filtered in the third stage to produce accurate secondary structures. The new release of PSIPRED2 achieved 78% accuracy on average.

VISSA (Visualization of Secondary Structure elements of Improving multiple Alignments) provides a good color-oriented visualization of predicted secondary structures to check for the consistency between multiple sequence alignment features and the secondary structures [114]. The VISSA technique consists of data processing and visualization. In data processing CLUSTALW is used for generating the MSAs and the secondary structures are predicted for each sequence using PSIPRED. An XML file is generated with the alignments, predicted secondary structures, metadata, and confidence values. This dissertation uses PSIPRED and VISSA [114] for secondary structure predictions and visualization to verify the new domain models generated by the automated tool chains that are described in the third chapter.

All the above sections describe the background relating to the problem addressed in the dissertation. The next few sections describe the algorithmic and architectural advancements to reduce the computation time.

2.6 Algorithmic and Architectural Accelerators of BLAST and HMMER

Different algorithmic and architectural approaches were used to speedup the BLAST and HMMER tools; these are described in the following sections. Three levels of parallelism exist for large batch BLAST processing: fine grained, medium grained, and coarse grained. In fine-grained parallelism, the comparisons of alignments are done in parallel. One input query sequence is aligned with one target sequence of the database, and the alignments of the comparison are done concurrently. In medium grained parallelism the database is partitioned into fragments, and one input query sequence is aligned with multiple target sequences of each database fragment in parallel. In coarse-grained parallelism, the database is replicated and multiple input query sequences are independently processed using this replicated database [19,20].

The nucleotide and protein sequences in various databases are growing at an exponential rate [26], and doubling their size every six months [40], but according to Moore's law [27] the number of transistors on a chip double every 18 months. Processor performance improvements are not keeping up with the growth of sequence databases. This led to porting of the BLAST and HMMER algorithms onto supercomputers, clusters of computers, shared memory architectures, and also network of workstations. Parallel virtual machine (PVM) libraries [19] along with message passing interface (MPI) [23] libraries and Linda [19, 20] were used to speed up both BLAST and HMM searches.

2.6.1 Algorithmic Speedups

The past decade, many parallel approaches of BLAST were developed such as TurboBLAST [21], Hyper-BLAST [22], mpiBLAST [23], BLAST services on OBIGrid [25], and ScalaBLAST [26]. There are advantages and limitations with each of these different parallel BLAST algorithms. NCBI has developed their own multithreaded version of BLAST on shared-memory multiprocessor architectures [21]. This multithreaded version does not scale up very well because of the bandwidth limitations of the number of processors on a bus in shared-memory multiprocessor. TurboBLAST addresses the problem of multithreaded BLAST by coordinating the use of multiple copies of serial BLAST applications. TurboBLAST uses networked clusters of heterogeneous personal computers or workstations for multiple copies of serial BLAST to provide results similar to NCBI BLAST. TurboBLAST is also portable to parallel supercomputers and worldwide computing grids [21]. The developers of TurboBLAST claim a speedup of 16X with 11 nodes and a speedup of 14X on 8 nodes for two different data sets [21]. On the other hand Hyper-BLAST claims to overcome the limitations of inter-node parallelism by logically partitioning the database, proper initiation, and the coordination of communication protocols of BLAST used on the remote node. The developers of Hyper-BLAST claim to achieve a speedup of 12X on a 2-way 8-node cluster [22].

Since BLAST is both computationally intensive and embarrassingly parallel for independent BLAST sequence searches, MPI is used by mpiBLAST to parallelize BLAST [23]. In mpiBLAST the database is divided into fragments. Each node searches a smaller portion of the database, as the communication demands are not heavy. This database segmentation reduces the overhead of disk I/O and intercommunication between nodes achieving good speedups [23]. The authors of mpiBLAST [23] claim achieving near linear speedups of BLAST in most cases and super-linear speedups in low memory cases for hundreds of nodes. The pioBLAST is an optimized version of mpiBLAST, which allows flexible database partitioning using caching techniques, enabling parallel I/O on shared files and performing scalable result processing protocols [24].

High-throughput GRIDBLAST services are provided by OBIGrid [25]. The GRIDBLAST system consists of a query splitter, job dispatcher, task manager, results collector, and formatter. It uses servers and heterogeneous remote worker nodes, with different BLAST implementations and job schedulers for massive batch processing.

The parallel approaches described above are scalable from tens to hundreds of nodes but cannot handle thousands of nodes [26]. Hence the developers of a new parallel approach called ScalaBLAST [26] claims it scales linearly to thousands of processors on both distributed memory and shared memory architectures. ScalaBLAST adopted the features of previously implemented

parallel BLAST algorithms such as distributing the target databases across available memory, multi-level parallelism, parallel I/O, and latency hiding techniques through data pre-fetching, and effective task scheduling, for huge batch processing inputs [26].

2.6.2 Architectural Speedups

To speed up sequence analysis and protein domain identification, different types of computer architectures and custom-built hardware architectures are used. This dissertation targets job level parallelism for PSI-BLAST. Since there is no dependency between two PSI-BLAST sequence searches, and cluster computing is a cost effective solution to speedup such applications [41]. Supercomputers such as Blue Gene/L are needed for applications that require millions of BLAST searches per day. A Blue Gene/L system comprised of 4096 nodes with dual 700 MHz PowerPC 440d processors is capable of performing 2 million BLAST searches a day against the nr (non-redundant) database that had 2.5 million-protein sequences at that time, achieving good speedups and efficiency [42]. BLAST searches are memory, bandwidth, and I/O intensive programs [43, 44]. On shared memory processor systems, BLAST uses threads to achieve parallelism [43]. In shared memory processor architectures bus bandwidth can be a problem if all the processors share the same memory bus. Specialized hardware architectures used to speedup BLAST include Processor

In Memory (PIM) [44], and Field Programmable Gate Arrays (FPGAs) [45, 121, 122, 123].

Both coarse-grained and fine-grained parallelism is exploited to design specialized hardware architectures to accelerate HMMER algorithms. HMMER is a computationally intensive algorithm, so higher the processor speed the faster the execution [46]. Hyper-threading and load balancing play significant roles in increasing speedups of HMMER [46]. JackHMMER exploits the coarse-grained parallelism to accelerate the profile-HMM searches [47]. JackHMMER [47] is a version of HMMER designed to run on an Intel IXP 2850 network processor that consists of heterogeneous multi-core processors. It uses a high degree of thread level parallelism on network processors, which outperforms the hyper-threaded HMMER version on Pentium 4. ClawHMMER, a streaming algorithm written in the Brook language, is a version of HMMER designed to run on graphics processors outperforms CPU implementations by many folds shown in table 2 of the paper [48]. FPGAs are used to design an accelerator for HMM search that exploits both coarse-grained and fine-grained parallelism [49]. FPGA-based hardware accelerator of HMM search achieved 100-fold speedup over the software HMM search implementation [49]. Opteron processors are also used to accelerate HMMER searches with minimally invasive recoding, and the authors claim to achieve better performances than Intel architectures [50]. This dissertation will take advantage of available accelerated algorithms and architectures and use them to accelerate our automated tool chain for protein

domain modeling. The next chapter describes the automated tool chain design for addressing the protein domain-modeling problem.

Chapter Three

Automated Tool Chain Design

There are three different aspects of protein domain modeling: domain identification, domain verification, and domain discovery. Hence for the domain identification process we used PSI-BLAST to get all the immediate relatives of the query unknown region, and then performed a HMMER search on the resultant similarity matched sequences. The HMMER tool identified the domain models for some immediate relatives of the query unknown regions. This shows the domain models in Pfam database are not sensitive enough to consider all plausible relatives of an unknown region. Hence tools currently in use such as HMMER are not sensitive enough by themselves to identify all plausible domains for a specific unknown region from the given Pfam domain models.

Thus the combination of the PSI-BLAST and HMMER tools are explored to search deeper and get all probable relatives, even more distant ones, of the query unknown region and check whether HMMER identifies any domains for these sequences. For domain verification process one has to perform exhaustive PSI-BLAST searches on every matching unknown region till no new matching proteins are acquired. The exhaustive PSI-blast process is an attempt to define the entire space of related sequences in the nr database, which is tedious and time consuming. Hence we took ten matched peptide sequences from the PSI-BLAST resultant file, picked five from the middle and five from the bottom of the

resultant proteins. These ten sequences are the matching regions of the resultant proteins that are similar to our query unknown region but not the entire protein sequence. Now PSI-BLAST is run on these 10 sequences individually that will result in most of the related sequence space. HMMER tool is used to identify the domains for these resultant proteins from the PSI-BLAST searches and check any domains existed in these similarity regions. Again some domains were identified in these distant relatives that were missed before. Thus concluding the domain models in the Pfam database are not good enough to identify domains for these distant relatives. By domain verification process, we make sure that no domain knowledge is currently present for the query unknown regions for which no domains are found. These peptide regions are used for discovering new protein domain models.

This led to the development of a tool chain that is sensitive enough to find all the domains that are likely for a protein, also considering related proteins to that specific query protein. PSI-BLAST is a sensitive searching tool that identifies all probable related proteins or relatives that matched the specific query protein. Then *hmmpfam* is run on all the related proteins from PSI-BLAST to identify the domains. This process is performed manually, which takes a considerable amount of time per query sequence, and currently there are no publicly available tools to automate this process. This led to the development of the automated tool chains that combine the PSI-BLAST and HMMER tools for domain identification and domain verification that are described in the sections 3.1 and 3.2.

Once the domains of all proteins of a specific genome are identified using *hmmpfam* search using the Pfam database by the automated tool chain. There are many peptide regions in these proteins for which no domains are identified but have potential regions for new domains. This resulted in the development of a new automated tool chain that can be used for domain discovery. We retrieve all the peptide regions for which no domains are identified based on the existing domain model databases using the domain identification and domain verification tool chains. This new automated tool chain uses a combination of the MUSCLE and HMMER tools to discover new domain models. First all the sequence similarity regions resulting from the PSI-BLAST search are retrieved. Then the MUSCLE tool is used to generate the multiple sequence alignments. The HMMER tools are used to generate the new domain models and are also used for verification of these newly generated models. The automated tool chain used for domain discovery is explained in the section 3.3.

3.1 Domain Identification Automated Tool chain (DIAT)

This section describes different tools used in this automated tool chain along with improvements implemented to reduce the computation time. Two different tools of the NCBI BLAST suite, one tool of the HMMER package, and resultant data file manipulation tools are used to design this automated tool chain. First, all the

unknown regions for which there are no domains identified by HMMER tool for a specific genome are extracted. Shewanella and E.coli genomes were used as the test bench genomes for this dissertation. All the unknown regions that do not have any domains are extracted from the MIST database. Only the unknown regions greater than 80 AA are considered for further analysis, as the smaller domains averages around 100 AA in size. For these peptides, the BLAST input generator tool (as shown in Figure 3.1) generates a fasta format file known as PSIIN as shown in Figure 3.2. These fasta files are inputs to the automated tool chain. Now the *blastpgp* tool is used to get all the closest relatives of the query unknown region from the protein database as shown in Figure 3.3. The various parameters used for running the *blastpgp* tool are described below.

The database used for searching is the nr (non-redundant) protein database (Figure 3.4). Four PSI-BLAST iterations are sufficient for sensitive homology searching, and any more iterations may lead to profile wander [29, 52]. The output file can be generated in different file formats such as text, XML, and ASCII. Two parsing tools are written one for parsing the tab delimited BLAST output and another for XML BLAST output.

The PSI-BLAST tool generates an output file known as PSIOUT as shown in Figure 3.5. PSIOUT is parsed using the BLAST results parsing tool and all the proteins that pass the threshold of 0.001 are extracted to reduce the false

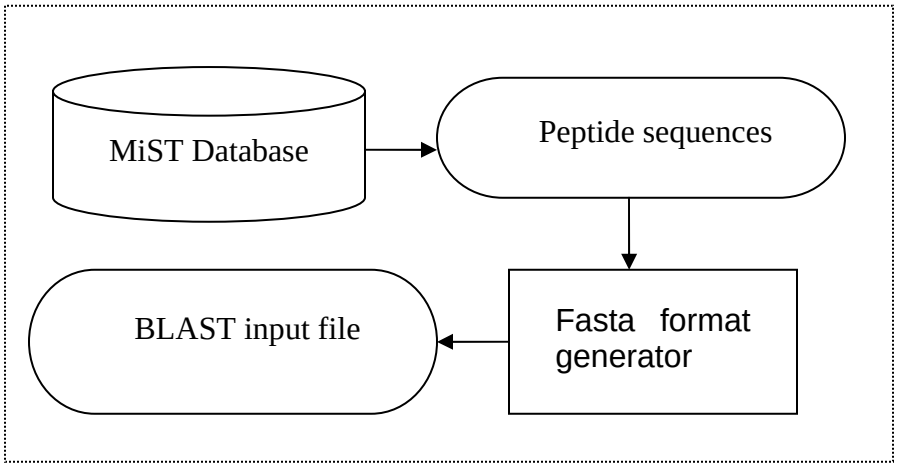


Figure 3.1: BLAST input generator

>24371603_342-457
LTGENLEMTEEKGYSVYRISAKTGLGVDELKQHLKSLMGYQSNLEGGFIARRR
HLEALEIAASHLQLGKEQLEVYLAGELLAEELRMAQLALSEITGRFTSDDLLGKIF
SSFCIGK

Figure 3.2: Input query (unknown region) to PSI-BLAST.

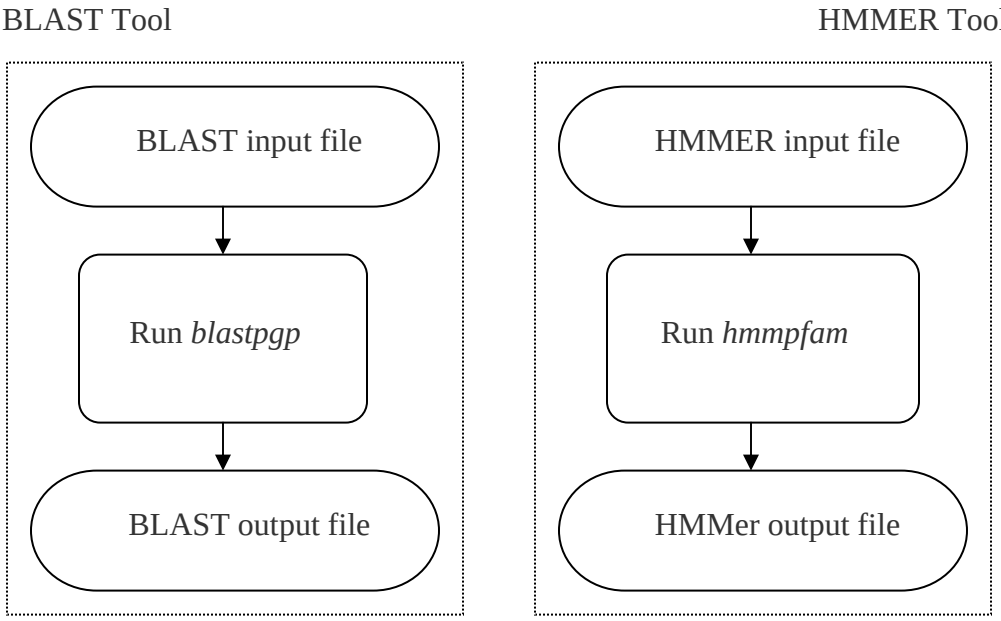


Figure 3.3: The core modules of the DIAT

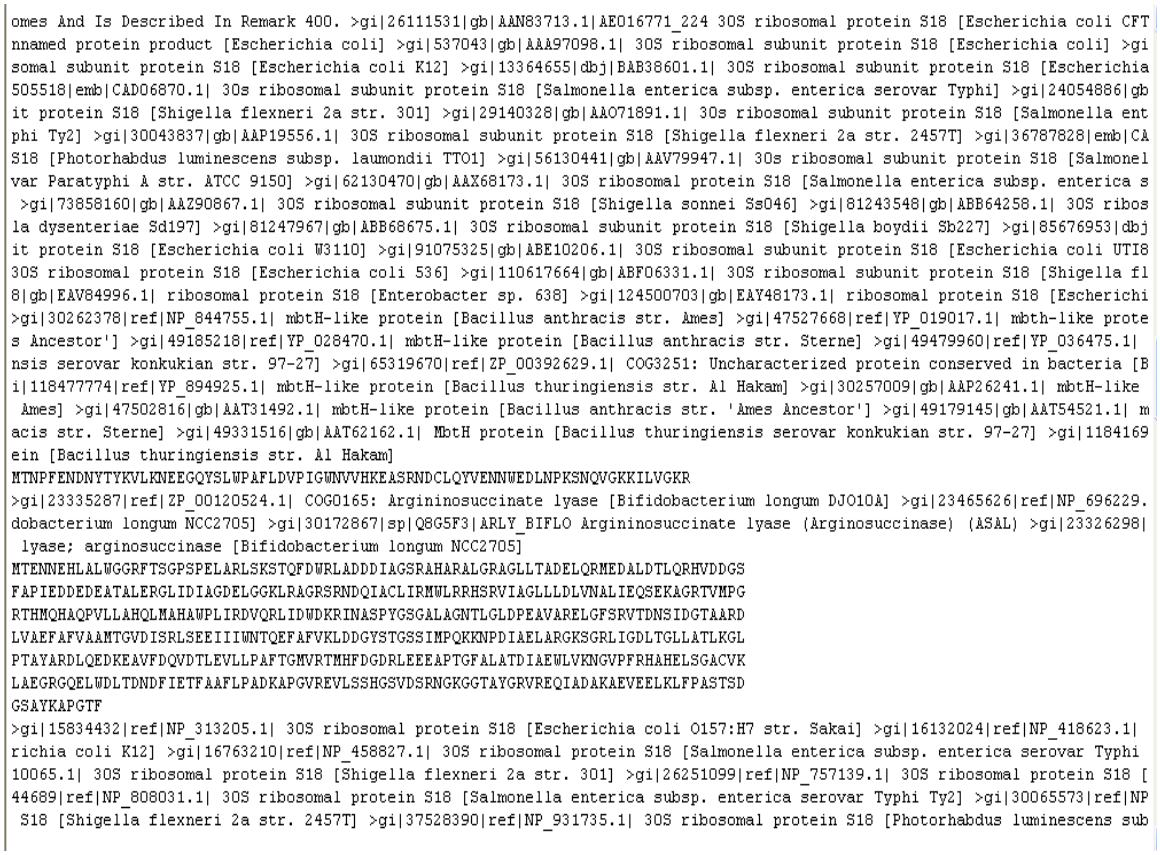
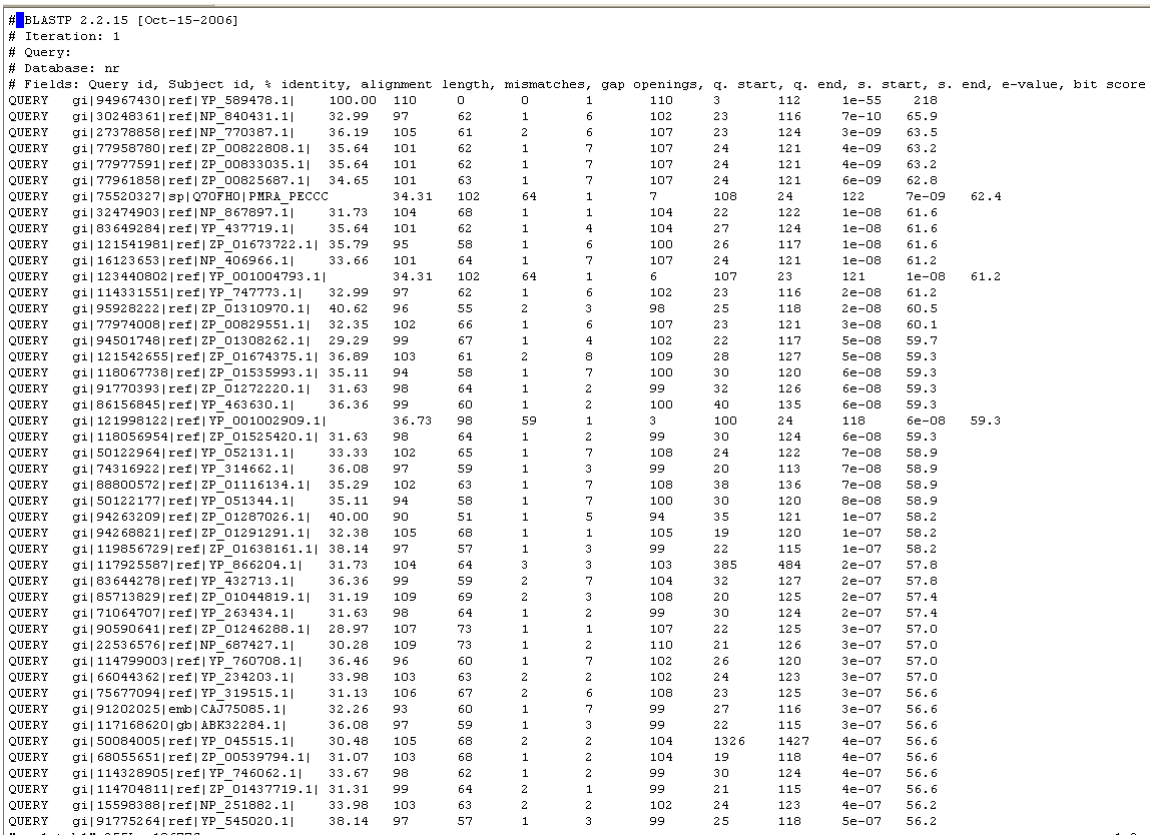


Figure 3.4: Screenshot of section of nr protein database



```
#BLASTP 2.2.15 [Oct-15-2006]
# Iteration: 1
# Query:
# Database: nr
# Fields: Query id, Subject id, % identity, alignment length, mismatches, gap openings, q. start, q. end, s. start, s. end, e-value, bit score
QUERY   gi|94967430|ref|YP_589478.1|    100.00  110     0       0       1       110     3       112     1e-55    218
QUERY   gi|30248361|ref|NP_840431.1|    32.99   97      62      1       6       102     23      116     7e-10   65.9
QUERY   gi|27378858|ref|NP_770387.1|    36.19   105     61      2       6       107     23      124     3e-09   63.5
QUERY   gi|77958780|ref|ZP_00822808.1|  35.64   101     62      1       7       107     24      121     4e-09   63.2
QUERY   gi|77977591|ref|ZP_00833035.1|  35.64   101     62      1       7       107     24      121     4e-09   63.2
QUERY   gi|77961858|ref|ZP_00825687.1|  34.65   101     63      1       7       107     24      121     6e-09   62.8
QUERY   gi|75520327|sp|Q7CFH0|PMRA_PECCC        34.31   102     64      1       7       108     24      122     7e-09   62.4
QUERY   gi|32474903|ref|NP_867897.1|    31.73   104     68      1       1       104     22      122     1e-08   61.6
QUERY   gi|83649284|ref|YP_437719.1|    35.64   101     62      1       4       104     27      124     1e-08   61.6
QUERY   gi|121541981|ref|ZP_01673722.1| 35.79   95      58      1       6       100     26      117     1e-08   61.6
QUERY   gi|16123653|ref|NP_406966.1|    33.66   101     64      1       7       107     24      121     1e-08   61.2
QUERY   gi|123440802|ref|YP_001004793.1|        34.31   102     64      1       6       107     23      121     1e-08   61.2
QUERY   gi|114331551|ref|YP_747773.1|   32.99   97      62      1       6       102     23      116     2e-08   61.2
QUERY   gi|95928222|ref|ZP_01310970.1|  40.62   96      55      2       3       98      25      118     2e-08   60.5
QUERY   gi|77974008|ref|ZP_00829551.1|  32.35   102     66      1       6       107     23      121     3e-08   60.1
QUERY   gi|94501748|ref|ZP_01308262.1|  29.29   99      67      1       4       102     22      117     5e-08   59.7
QUERY   gi|121542655|ref|ZP_01674375.1| 36.89   103     61      2       8       109     28      127     5e-08   59.3
QUERY   gi|118067738|ref|ZP_01535993.1| 35.11   94      58      1       7       100     30      120     6e-08   59.3
QUERY   gi|91770393|ref|ZP_01272220.1|  31.63   98      64      1       2       99      32      126     6e-08   59.3
QUERY   gi|86156845|ref|YP_463630.1|    36.36   99      60      1       2       100     40      135     6e-08   59.3
QUERY   gi|121998122|ref|YP_001002909.1|        36.73   98      59      1       3       100     24      118     6e-08   59.3
QUERY   gi|118056954|ref|ZP_01525420.1| 31.63   98      64      1       2       99      30      124     6e-08   59.3
QUERY   gi|50122964|ref|YP_052131.1|    33.33   102     65      1       7       108     24      122     7e-08   58.9
QUERY   gi|74316922|ref|YP_314662.1|    36.08   97      59      1       3       99      20      113     7e-08   58.9
QUERY   gi|88800572|ref|ZP_01116134.1|  35.29   102     63      1       7       108     38      136     7e-08   58.9
QUERY   gi|50122177|ref|YP_051344.1|    35.11   94      58      1       7       100     30      120     8e-08   58.9
QUERY   gi|94263209|ref|ZP_01287026.1|  40.00   90      51      1       5       94      35      121     1e-07   58.2
QUERY   gi|94268821|ref|ZP_01291291.1|  32.38   105     68      1       1       105     19      120     1e-07   58.2
QUERY   gi|119856729|ref|ZP_01638161.1| 38.14   97      57      1       3       99      22      115     1e-07   58.2
QUERY   gi|117925587|ref|YP_866204.1|   31.73   104     64      3       3       103     385     484     2e-07   57.8
QUERY   gi|83644278|ref|YP_432713.1|    36.36   99      59      2       7       104     32      127     2e-07   57.8
QUERY   gi|85713829|ref|ZP_01044819.1|  31.19   109     69      2       3       108     20      125     2e-07   57.4
QUERY   gi|71064707|ref|YP_263434.1|    31.63   98      64      1       2       99      30      124     2e-07   57.4
QUERY   gi|90590641|ref|ZP_01246288.1|  28.97   107     73      1       1       107     22      125     3e-07   57.0
QUERY   gi|22536576|ref|NP_687427.1|    30.28   109     73      1       2       110     21      126     3e-07   57.0
QUERY   gi|114799003|ref|YP_760708.1|   36.46   96      60      1       7       102     26      120     3e-07   57.0
QUERY   gi|66044362|ref|YP_234203.1|    33.98   103     63      2       2       102     24      123     3e-07   57.0
QUERY   gi|75677094|ref|YP_319515.1|    31.13   106     67      2       6       108     23      125     3e-07   56.6
QUERY   gi|91202025|emb|CAJ75085.1|     32.26   93      60      1       7       99      27      116     3e-07   56.6
QUERY   gi|117168620|gb|ABK32284.1|     36.08   97      59      1       3       99      22      115     3e-07   56.6
QUERY   gi|50084005|ref|YP_045515.1|    30.48   105     68      2       2       104     1326    1427    4e-07   56.6
QUERY   gi|68055651|ref|ZP_00539794.1|  31.07   103     68      1       2       104     19      118     4e-07   56.6
QUERY   gi|114328905|ref|YP_746062.1|   33.67   98      62      1       2       99      30      124     4e-07   56.6
QUERY   gi|114704811|ref|ZP_01437719.1| 31.31   99      64      2       1       99      21      115     4e-07   56.6
QUERY   gi|15598388|ref|NP_251882.1|    33.98   103     63      2       2       102     24      123     4e-07   56.2
QUERY   gi|91775264|ref|YP_545020.1|    38.14   97      57      1       3       99      25      118     5e-07   56.2
```

Figure 3.5: Tab-delimited PSI-BLAST output file (PSIOUT)

positives [2, 29]. The resulting PROTOUT file from the parsing tool consists of the protein identification number in the GenBank Identification number along with the starting and ending AA sequence numbers representing the region of the similarity match with the query sequence. The entries in the PROTOUT file are sorted in assending order based on e-value.

Based on prior knowledge and the design of the automated tool chain, a database was created to save computation time. A reference database known as refDB with two tables, refProteins and refDomains, is created. The refProteins table consists of the ID and name for each new protein resulting from PSI-BLAST searches. The refDomains table consists of all the domain information resulted from hmmpfam searches, such as domain name, the starting and ending AA sequence numbers representing the domain region and protein name that resulted in the above domain. These two tables are used as a database to house the information of every new protein hit from the DIAT tool chain.

Once the results from PSI-BLAST are obtained, the Hmmer input generator tool checks the protein IDs against the refDB database to see whether there are any hits in the database as shown in Figure 3.6. Only the new protein hits that are not present in refProteins table are used to generate the input file for the HMMER tool. For the proteins that are found in the refDB, domain information is retrieved from the refDomains table for further analysis.

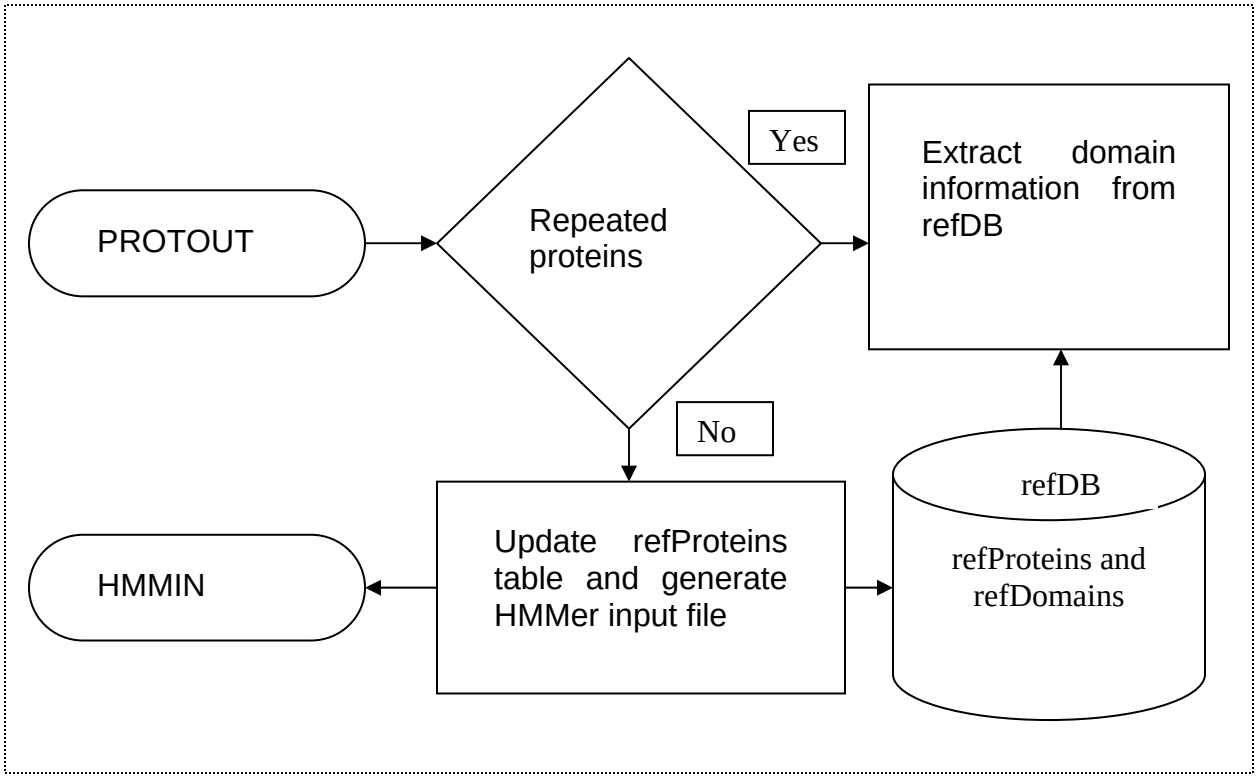
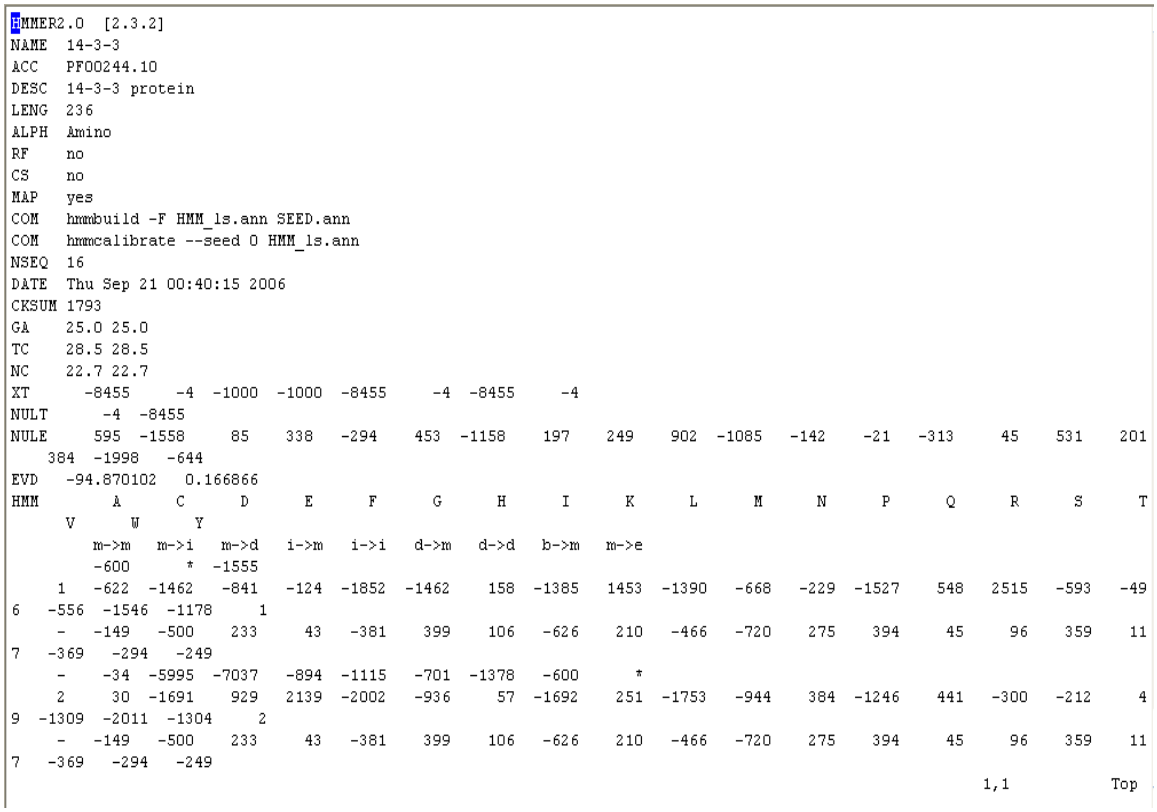


Figure 3.6: HMMER input generator

Next *db2fasta* is used to extract the fasta sequences for all the proteins from the nr database and populate a fasta file known as HMMIN by the Hmmer input generator tool. This file is used as the input file for the HMMER tool. The *hmmpfam* of HMMER tool is used for the domain identification, which outputs all the domains possible for the proteins in HMMIN using the Pfam database shown in Figure 3.3. The *hmmpfam* tool evaluates whether a match is significant or not and outputs the domains with significant matches using the Pfam database (Figure 3.7).

The *hmmpfam* output is a huge text file known as HMMOUT with all the proteins and their significant domains as shown in Figure 3.8. The HMMER results parser tool is used to parse HMMOUT and output the results into a file known as DOMOUT1 with protein name, domain names, and the starting and ending AA sequence numbers representing the domain region. The data from the DOMOUT1 file is used to populate the refDomains table of the refDB database. Once the results from the HMMER tool are generated, the new domain information for the new protein hits along with the domain information of the repeated proteins from the refDomains table is used to generate the DOMOUT2 file. DOMOUT2 file consists of the entire domain information for all the proteins in the PROTOUT file for the respective query unknown region. As the number of proteins in the refDB database increases, the chances of finding protein hits from the database tables increases, thus reducing the computational time drastically by eliminating the need for HMMER reevaluations.



```
HMMER2.0  [2.3.2]
NAME  14-3-3
ACC   PF00244.10
DESC  14-3-3 protein
LENG  236
ALPH  Amino
RF    no
CS    no
MAP   yes
COM   hmmbuild -F HMM_ls.ann SEED.ann
COM   hmmcalibrate --seed 0 HMM_ls.ann
NSEQ  16
DATE  Thu Sep 21 00:40:15 2006
CKSUM 1793
GA    25.0 25.0
TC    28.5 28.5
NC    22.7 22.7
XT      -8455     -4  -1000  -1000  -8455     -4  -8455     -4
NULT      -4  -8455
NULE     595  -1558     85    338   -294    453  -1158    197    249    902  -1085   -142    -21   -313     45    531    201
    384  -1998   -644
EVD   -94.870102   0.166866
HMM        A      C      D      E      F      G      H      I      K      L      M      N      P      Q      R      S      T
      V      W      Y
         m->m   m->i   m->d   i->m   i->i   d->m   d->d   b->m   m->e
         -600      *  -1555
     1   -622  -1462   -841   -124  -1852  -1462    158  -1385   1453  -1390   -668   -229  -1527    548   2515   -593   -49
6   -556  -1546  -1178     1
     -   -149   -500    233     43   -381    399    106   -626    210   -466   -720    275    394     45     96    359     11
7   -369   -294   -249
     -    -34  -5995  -7037   -894  -1115   -701  -1378   -600      *
     2     30  -1691    929   2139  -2002   -936     57  -1692    251  -1753   -944    384  -1246    441   -300   -212      4
9  -1309  -2011  -1304     2
     -   -149   -500    233     43   -381    399    106   -626    210   -466   -720    275    394     45     96    359     11
7   -369   -294   -249
                                                                                                                  1,1           Top
```

Figure 3.7: Screenshot of section of Pfam Database

```
hmmpfam - search one or more sequences against HMM database
HMMER 2.3.2 (Oct 2003)
Copyright (C) 1992-2003 HHMI/Washington University School of Medicine
Freely distributed under the GNU General Public License (GPL)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
HMM file:                 /home/data2/bhanu/blast/toolkit/blast_demo/Pfam_ls
Sequence file:            /home/data2/bhanu/blast/toolkit/blast_demo/results/24371603_342-457Hmme
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Query sequence: gi|24371603|ref|NP_715645.1|
Accession:      [none]
Description:    tRNA modification GTPase [Shewanella oneidensis MR-1] >gi|32171819|sp|Q8CX52|TRME

Scores for sequence family classification (score includes all domains):
Model    Description                                   Score    E-value  N
-------- -----------                                   -----    ------- ---
MMR_HSR1 GTPase of unknown function                    134.8    2.3e-37   1
Miro     Miro-like protein                              31.4    3.1e-06   1

Parsed for domains:
Model    Domain  seq-f seq-t    hmm-f hmm-t      score  E-value
-------- ------- ----- -----    ----- -----      -----  -------
MMR_HSR1   1/1     220   341 ..     1   140 []   134.8  2.3e-37
Miro       1/1     221   341 ..     1   139 []    31.4  3.1e-06

Alignments of top-scoring domains:
MMR_HSR1: domain 1 of 1, from 220 to 341: score 134.8, E = 2.3e-37
                   *->lrvaivGrPNVGKSTLiNaLtgkkraiVsdyPGtTrdpnegrveldg
                      ++v+i+GrPN+GKS+L+NaL+gk+ aiV+ ++GtTrd+ + +++ldg
  gi|243716Q   220    MKVVIAGRPNAGKSSLLNALAGKESAIVTEIAGTTRDVLREHIHLDG 266

                   rqiilvDTPGiiegasegegelgertleaieeaDlilfVvDasegltedy
                    ++ ++DT+G++++ +++e++++er++ +i  aD +lf+vD++
  gi|243716Q   267 MPLHIIDTAGLRDTTDTVEQIGIERAWNEINSADRVLFMVDGTTTTA--- 313
```

Figure 3.8:Typical HMMER output file (HMMOUT)

Finally, the final results parser tool compares the outputs generated by the PSI-BLAST results parser tool (PROTOUT) and the HMMER results parser tool (DOMOUT2) to check whether there are any domains present in the similarity regions of the matching proteins and populates the FINALOUT file. The FINALOUT consists of the entire protein and domain information. If the tool chain identifies no domains, FINALOUT reflects this result. The flow diagram of the DIAT is shown in Figure 3.9. Once all FINALOUT files of all the unknown regions are generated, these files are parsed and the domains identified for the unknown regions of the entire genome are used to populate the PepDomDB database that is describe in the section 3.4.

3.2 Domain Verification Automated Tool chain (DVAT)

The same tool set used for DIAT is used in DVAT but is slightly modified for speculative domain discovery and this process is called domain verification. Two different tools of the NCBI BLAST suite, one tool of the HMMER package, and data file manipulation tools are used to design this automated tool chain. The input to DVAT is all the unknown regions for which no domains are identified by DIAT along with the PSIOUT files from DIAT of the respective query unknown region. In DIAT only the close relatives for the specific query unknown region are found but in DVAT all the possible relatives for the specific unknown region are found by repeating the PSI-BLAST on various protein hits obtained from the PSI-BLAST results of the DIAT tool chain.

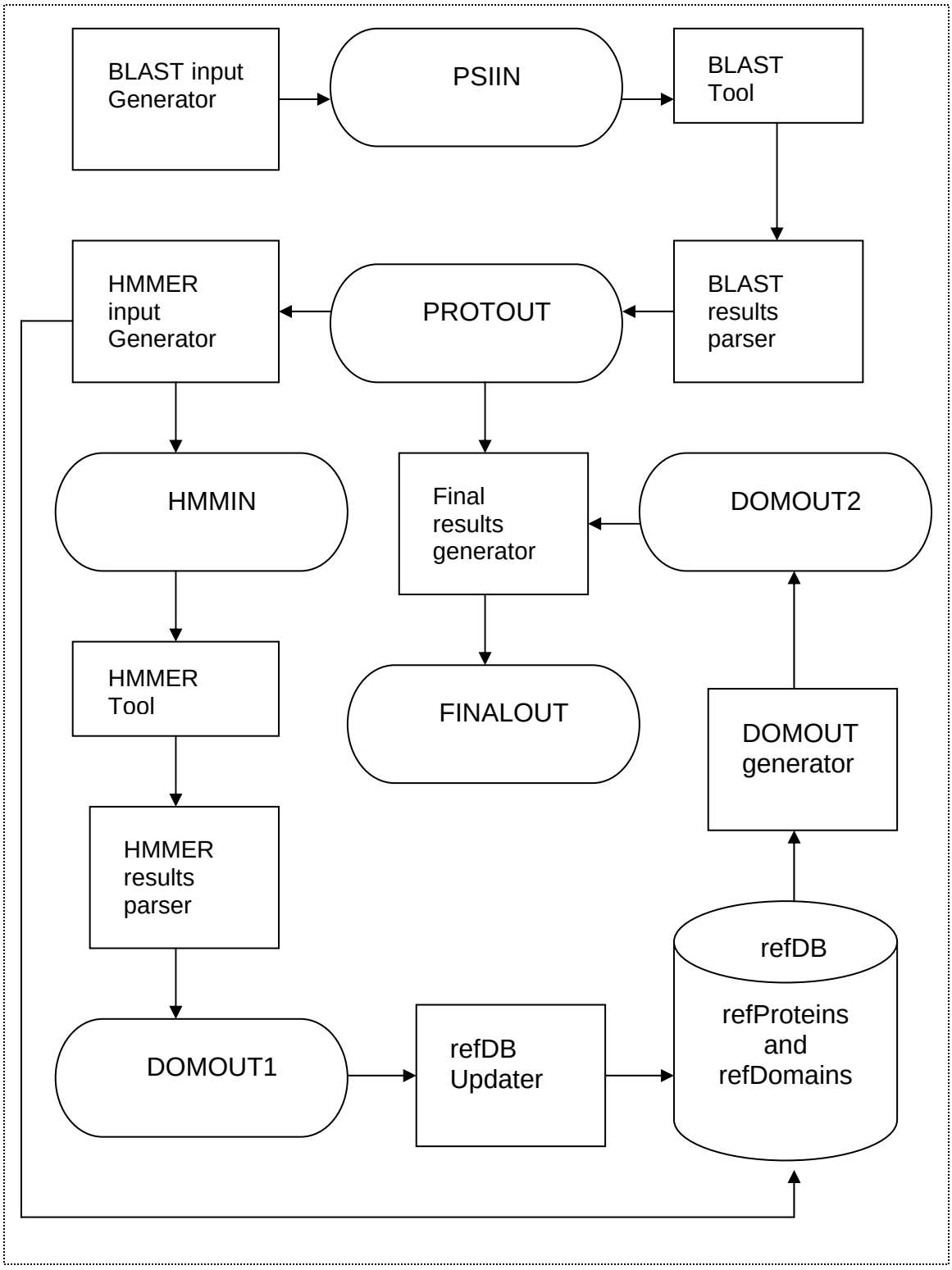


Figure 3.9: The Domain Identification Automated Tool chain flow

The flow diagram of DVAT is shown in Figure 3.10. The DVAT tool chain incorporates the DIAT approaches to save time. First ten proteins are picked from the PSI-BLAST resultant file of a specific unknown region. Five of these ten proteins are picked from the middle of the PROTOUT file and five from the end of the PROTOUT file. Then the similarity matching AA regions to the query unknown region are extracted for all these ten proteins from the fasta sequence file generated by the *db2fasta* tool from the nr database. Finally these ten matching sequence regions are used as the input to the PSI-BLAST searches instead of the entire fasta sequence of these similar proteins.

PSI-BLAST is run on all these ten matching sequence regions for four iterations. The PSIOUT files from the PSI-BLAST search are parsed and PROTOUT files are generated. The proteins from the PROTOUT files are first searched against the refDB database to check the repeated proteins. Only new proteins hits are used to generate the fasta sequence file that is used as the input to the HMMER tool. For all the repeated proteins, domain information is retrieved from the refDomains table of the refDB and the DOMOUT1 file is populated for further analysis. Next *hmmpfam* is run on all the new protein hits for domain verification, and the domain regions are compared to the matching peptide sequences to see whether there are any domains present in these regions similar to the DIAT results. Also the refDB is updated from the results generated from the HMMER process similar as in DIAT tool chain. A careful check is performed to match the results from *hmmpfam* to the correct peptide sequences, as there are ten

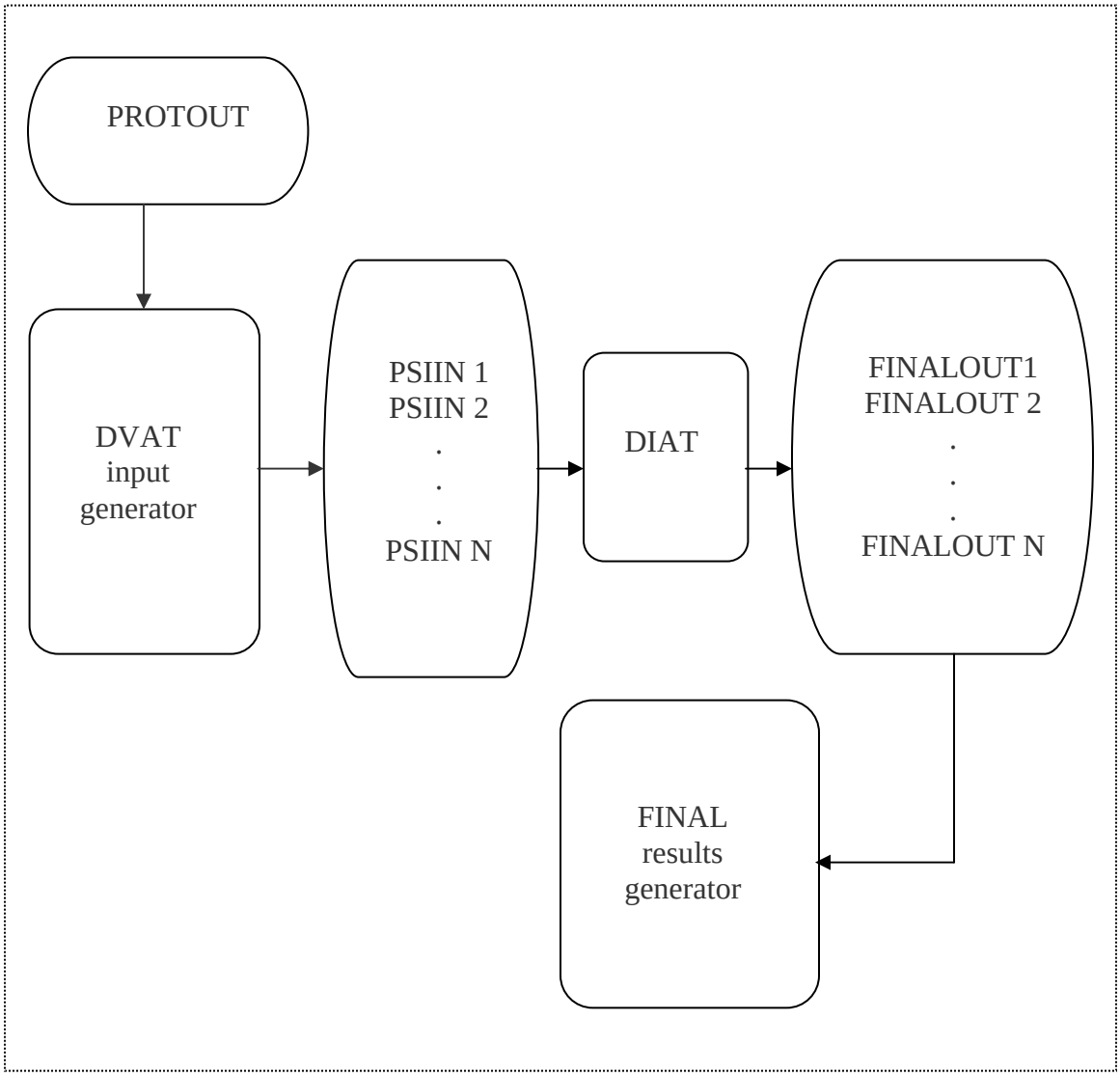


Figure 3.10: The Domain Verification Automated Tool chain flow

matching sequence regions here instead of one unknown region as in the DIAT tool for a query sequence. This process is repeated on all unknown regions of the input genome data file for which no domains were identified by the DIAT process.

If any domains are identified for either of these 10 matching peptide sequences, the DVAT tool chain will disregard this unknown region for speculative domain discovery. This means there are domains for the distant relative of the specific query sequence, and hence these unknown regions are not used for new domain discovery modeling. The rest of the unknown regions for which no domains are identified in any of the ten matching regions are the templates for speculative domains; these are used for the domain discovery process as described in the following section.

3.3 Domain Discovery Automated Tool chain (DDAT)

The domain discovery automated tool chain uses a different set of tools when compared to DIAT and DVAT for domain discovery process. We use the MUSCLE and HMMER tools along with resultant data file manipulation tools for domain discovery. The core modules used for building DDAT are shown in Figure 3.11. All sequences for which no domains are found in either the domain identification or domain verification processes, are used as inputs to the DDAT

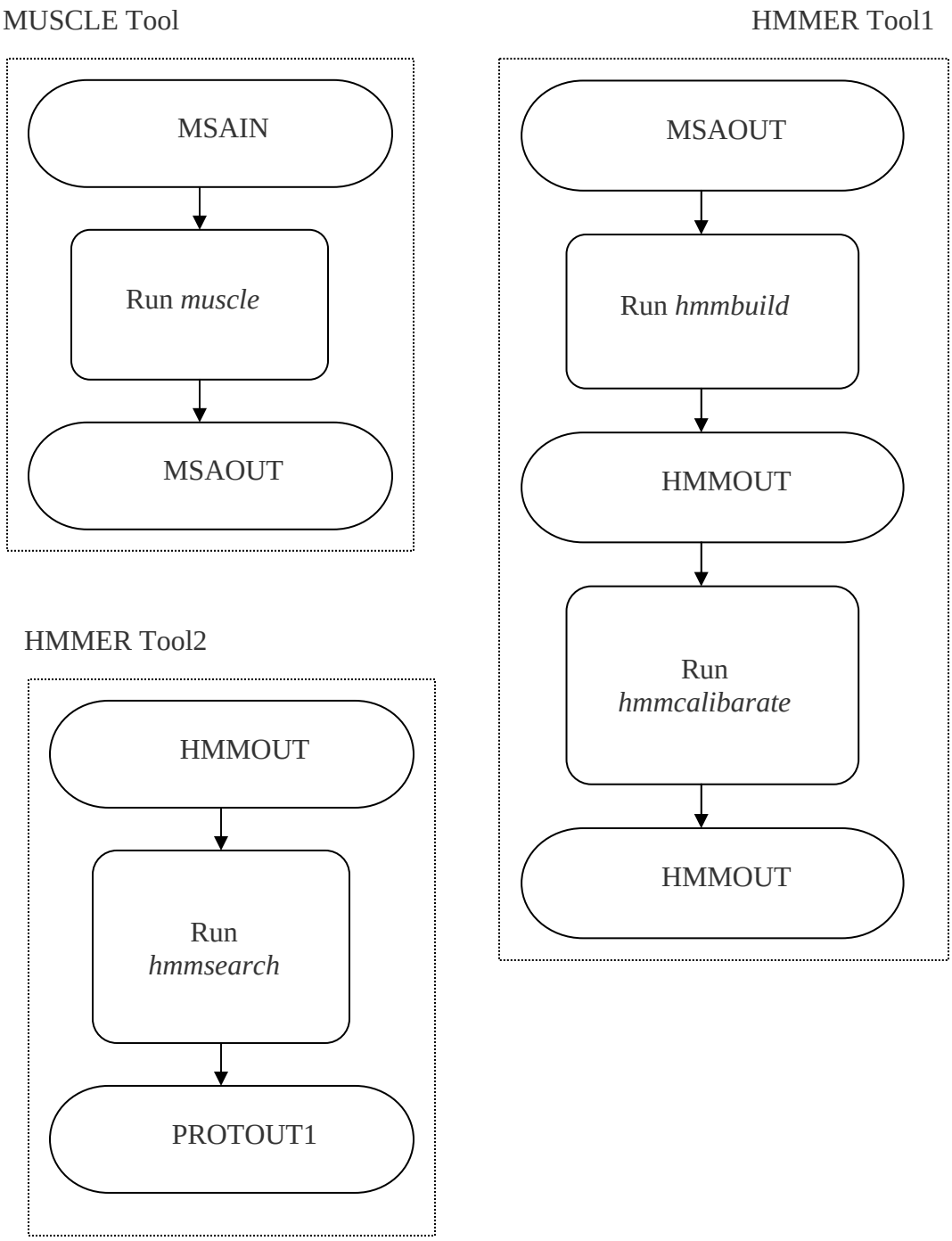


Figure 3.11: The core modules of the Domain Discovery Automated Tool chain

tool chain. First the MSA (Multiple Sequence Alignment) input generator retrieves all the resultant protein matches from the PSI-BLAST search for the specific query sequence. The input to this tool is the PROTOUT file generated by the DIAT tool chain. Then retrieving the amino acids sequence of the matched regions of the resultant similar proteins generates the MSAIN file. The flow diagram of the MSA input generator is shown in Figure 3.12.

As shown in Figure 3.11 the MSAIN file is fed to the MUSCLE tool for generating the multiple sequence alignments. The input to MUSCLE is either the amino acid or nucleotide sequences. The multiple sequence alignment file generated by MUSCLE is known as MSAOUT as shown in Figure 3.11. The MSAOUT file is inputted to the HMMER tool and *hmmbuild* is run on the multiple sequence alignments to build a HMM. This HMM is calibrated using *hmmcalibrate*. The HMM file is known as HMMOUT. HMMOUT is then fed into *hmmsearch* to search against the nr database for matching proteins. The resultant file from *hmmsearch* has proteins and matching regions similar to the PROTOUT file from the DIAT tool chain, hence it is called PROTOUT1 as shown in Figure 3.11.

A comparison of the protein results from *hmmsearch* and the proteins used to generate the MSAIN file is performed to check whether there are any additional proteins resulted from the *hmmsearch*. If no new proteins are found this indicates that the HMM model generated by the DDAT tool chain is a potential new domain model. If new proteins resulted from the *hmmsearch* are not in MSAIN proteins,

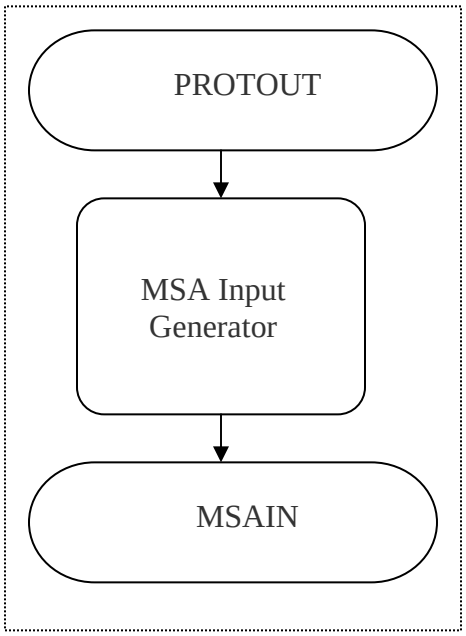


Figure 3.12: The multiple sequence alignment (MSA) input generator

this will result in further analysis of these new proteins. These additional proteins are then fed into HMMER tool and the *hmmpfam* along with the refDB is used to get the domains for these additional proteins. Then all the domain information is retrieved to build a DOMOUT2 file either by running *hmmpfam* on the new proteins or checking the refDB for existing proteins as shown in Figure 3.13. If the matching regions of these proteins correspond to an already existing domain model, this will imply that these are poor domain models in the Pfam database and need to be improved. And if the matching regions of these proteins do not result in any known domains, this implies that this HMM is a potential new domain model.

The Figure 3.13 shows the flow of the DDAT tool chain. The FINALOUT resulted from the tool chain will reflect whether the HMM model generated by the DDAT tool chain is either a new domain model or a poor existing model. These models are further analyzed by checking whether there are any secondary structures for these domain models. This output of the tool chain will give biologist a starting point to work on new domains rather than wasting time exploring the huge sequence space.

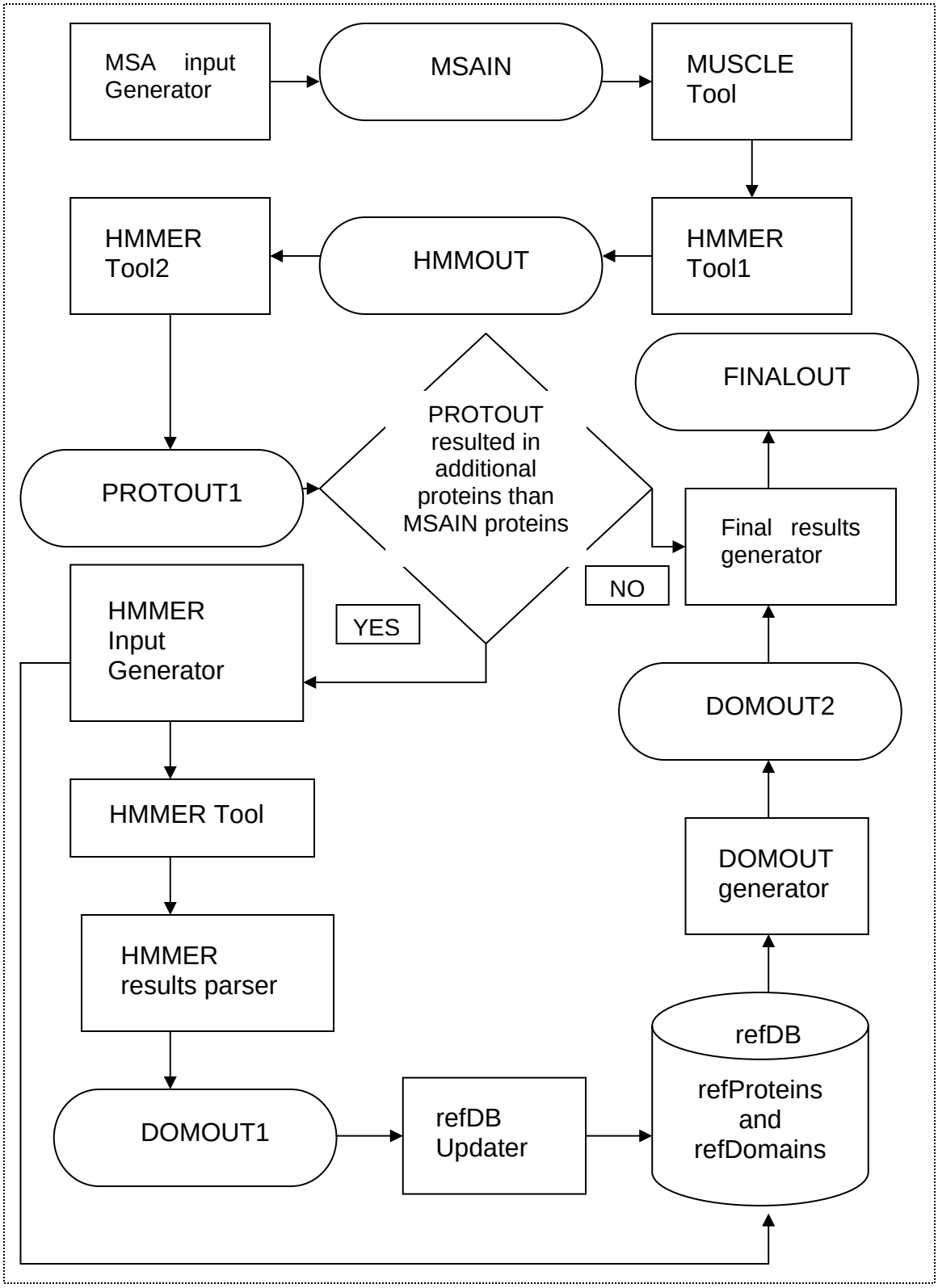


Figure 3.13: The Domain Discovery Automated Tool chain flow

3.4 PepDomDB Database

The PepDomDB database consists of the unknown regions for which domains were identified in the DIAT and DVAT tool chains. The database consists of the peptide id along with sequence information in PepInfo table. Domain information in DomInfo table consists of the respective peptide id to which this domain belongs, along with the domain name and domain sequence from and domain sequence to. This PepDomDB database can be used to further reduce the total computation time with a modified approach as follows. Whenever a new peptide sequence is sequenced. If the researcher wants to know the possible domains. One can search against the PepDomDB if the sequence matches 100 percent, then the domain information can be retrieved from the database. The computation time to retrieve information from the PepDomDB will be much smaller than the computation time required to run the DIAT and DVAT process on the peptide sequence. This PepDomDB database will be useful when it is made publicly available.

3.5 Domain Model Verification

The new domain models of Shewanella and E.coli genomes resulted from the unknown regions that are greater than 80 amino acids long and are between two known domains are used for further analysis. The secondary structures were

developed for these new domain models using VISSA [114] to check for the alpha helixes and beta sheets. Further these sequences were uploaded to PHYRE [117] (Protein Homology/analogY Recognition Engine) to check for the resemblance with the existing known domains or protein structures.

The next chapter describes the protein domain modeling results obtained by using Shewanella and E.coli genomes.

Chapter Four

New Domain Model Results

In this chapter we discuss the results obtained from the three phases of the genome-wide protein domain exploration using automated tool chains. Shewanella and E.coli genome sequences are used and the domain models for these two genomes are constructed. Figure 4.1 and 4.2 shows the different length distributions of the unknown sequences obtained from the Shewanella and E.coli genomes respectively. The automated tool chain generated new domain models for both the Shewanella and E.coli genomes, proving the robustness of the design.

The input sequences for the automated tool are unknown regions that pass the following filters:

a. The unknown regions are greater than or equal to 80 amino acids. This sequence length is small enough to identify smaller domains and long enough to avoid noise.
b. The unknown regions should not have any known domains, or coiled coil regions, or segments of low compositional complexity to reduce the computational time.
c. The unknown regions are extracted from the start of the protein sequence to the start of a domain, between domains, or between the end of a domain and to the end of the protein sequence as shown in the Figure 1.5.

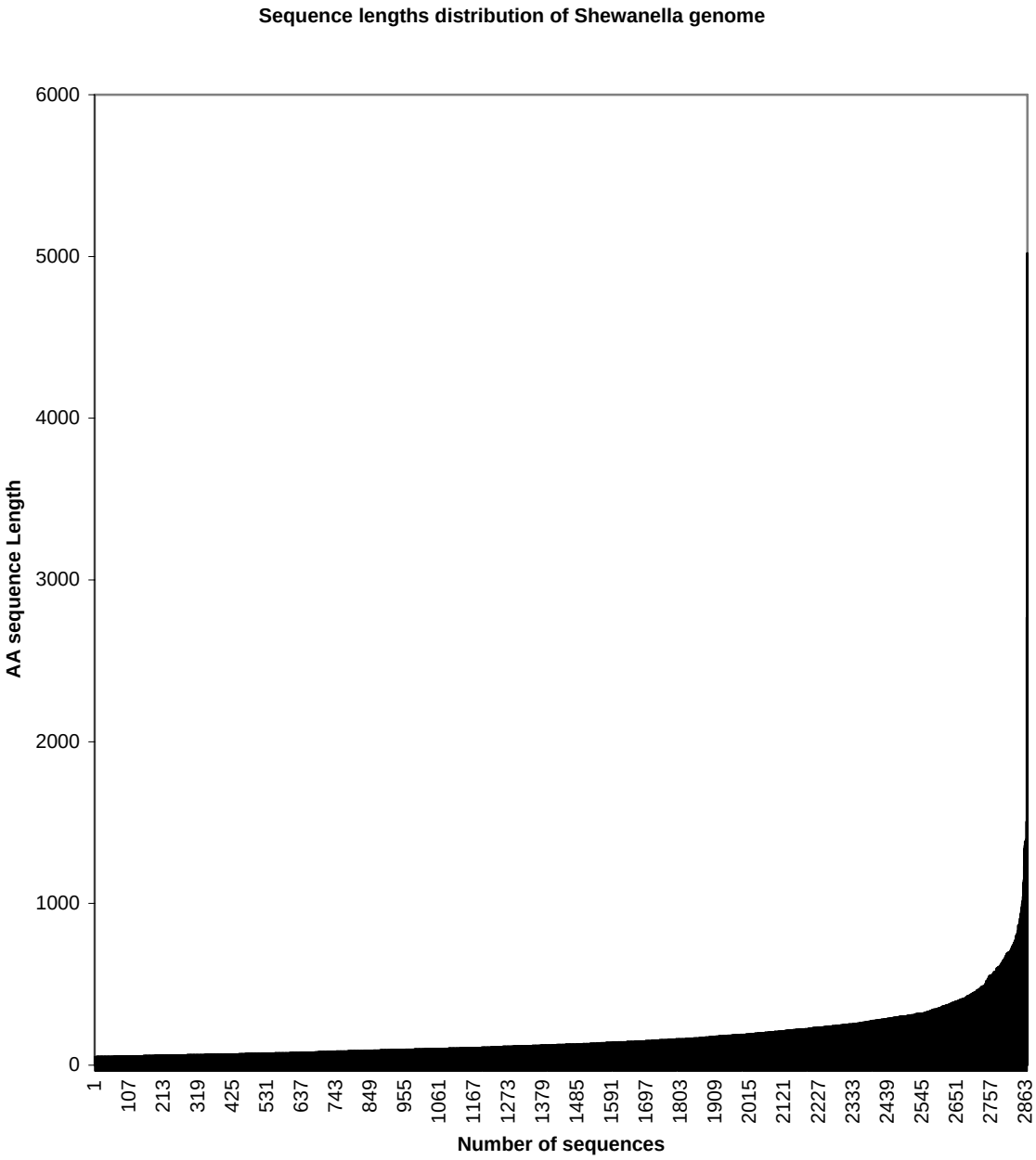


Figure 4.1: Sequence lengths distribution of Shewanella genome

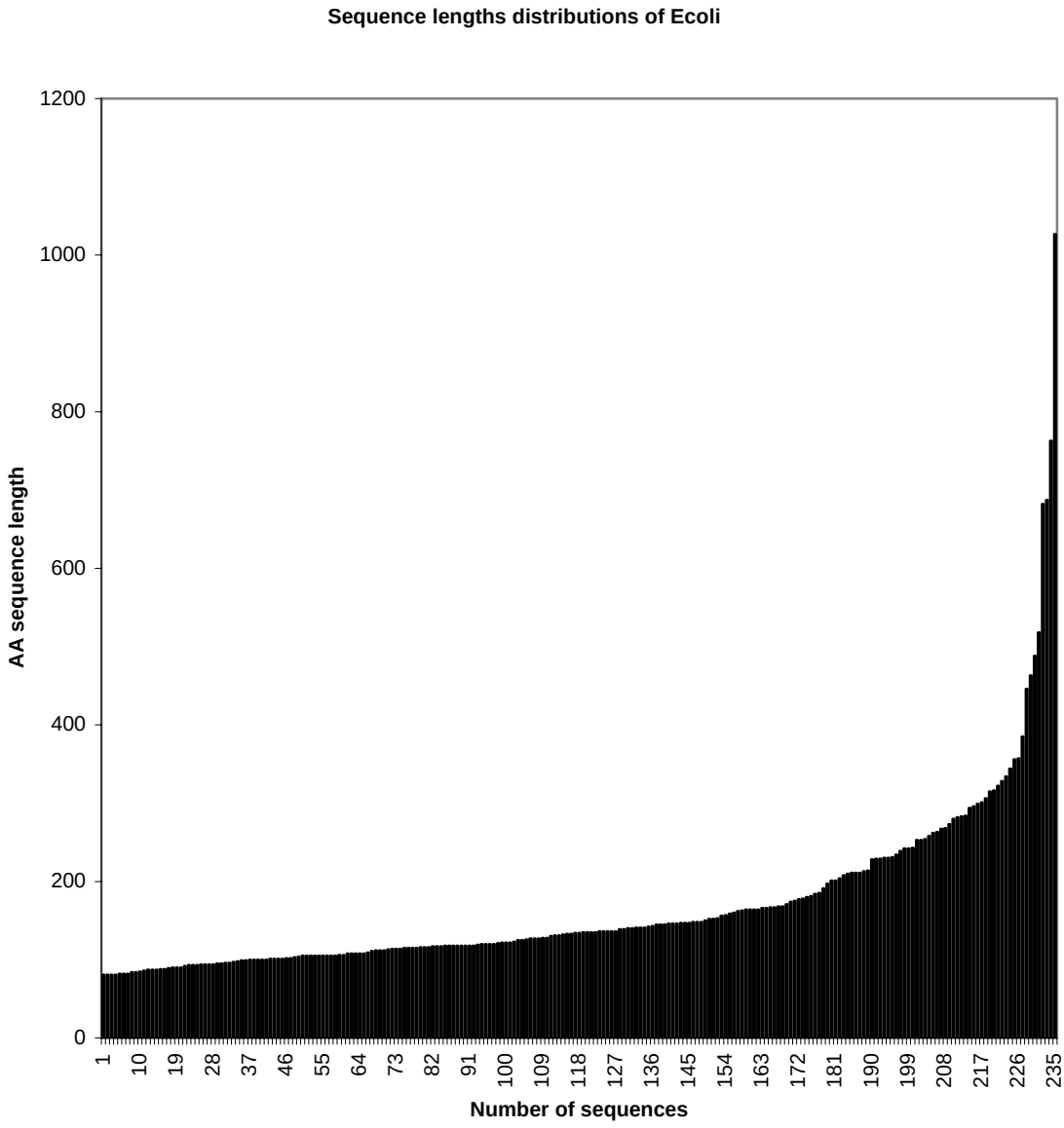


Figure 4.2: Sequence lengths distribution of E.coli genome

These sequences are then fed into the automated tool chains and new domain models are constructed at the end. The results obtained by individual phases of the domain exploration tool chain are discussed in the following sections.

4.1 Test-bench Files

Two test-bench organisms are used as inputs for the automated tool chains. Both Shewanella and E.coli are metabolically versatile bacteria [118, 119]. Shewanella has the ability to convert uranium dissolved in contaminated ground water to a non-soluble bi-product, preventing uranium contamination [119]. This ability of Shewanella makes it an important factor in cleaning up uranium when nuclear weapons are manufactured. Hence lot of effort is put into understanding Shewanella organism. The Shewanella genome has two input files, the first test bench has a total of 2867 unknown regions that passed the filters, denoted as the GENWIDEshew file used for protein domain modeling. The second test bench of 100 unknown regions of Shewanella genome for which the results were obtained manually, denoted as the MANGEN file. MANGEN is used to authenticate the working of the automated tool chains.

E.coli is able to grow in the presence and absence of oxygen thus making this organism important. E.coli is present in the lower intestines of mammals. E.coli assists with waste processing, vitamin K production and food absorption. E.coli

also causes diseases such as urinary tract infection, meningitis and pneumonia. Hence E.coli is one of the extensively studied organisms. There are 235 unknown regions that passed the filters for the E.coli genome denoted as the GENWIDEecoli used for protein domain modeling. Thus these two bacteria's make excellent test bench organisms.

4.2 DIAT Results

DIAT was run on 2867 unknown regions of the Shewanella genome and 235 unknown regions from the E.coli genome. E.coli is one of the most studied organisms and was sequenced much earlier than Shewanella; hence it resulted in fewer sequences with unknown domain regions than Shewanella even though both have around 5000 proteins in their genome. For the Shewanella genome PSI-BLAST was run for four iterations on the 2867 sequences, resulting in 576,010 total protein matches, out of which only 342,233 were unique proteins. This means that some proteins were found in more than one of the PSI-BLAST result files. These unique proteins are used to build the refProteins table of the refDB. Only these unique proteins are inputted into the HMMER tool. *Hmmpfam* is run on these unique proteins with all the significant domain results documented in the refDomains table of the refDB database. A total of 798,914 domains were identified for these 342,233 unique proteins, averaging a little more than 2 domains per protein.

The PSI-BLAST searches for the input sequences for E.coli genome resulted in a total of 61,921 protein matchs out of which 47,498 were unique proteins. Out of these 47,498 unique proteins there was no information in the refDB for only 10,436 proteins, hence only these new proteins were inputted to HMMER tool to get their domain information. Having the refDB database saved almost four fifths of the search time. These 10,436 proteins resulted in 32,729 domains, averaging around 3 domains per protein. The new proteins and domain information are added to the refDB database.

The domain information for all the matching regions of the protein sequences is retrieved from the refDB database. A check is performed to see whether there are any sequences for which domains are identified by the DIAT tool chain and the results are populated in the FINALOUTs. For the Shewanella genome out of 2867 input sequences domains were identified for 1664 sequences by DIAT tool chain. For the E.coli genome out of 235 input sequences domains were identified for 171 sequences by DIAT tool chain. The rest of the sequences 1203 for Shewanella and 64 for E.coli are used as inputs to the DVAT tool chain for further analysis. Figure 4.3 illustrates the DIAT results flow.

For the 1664 unknown regions of the Shewanella genome, a total of 3295 domains were identified including 1016 unique domains and their distribution is shown in Figure 4.4.

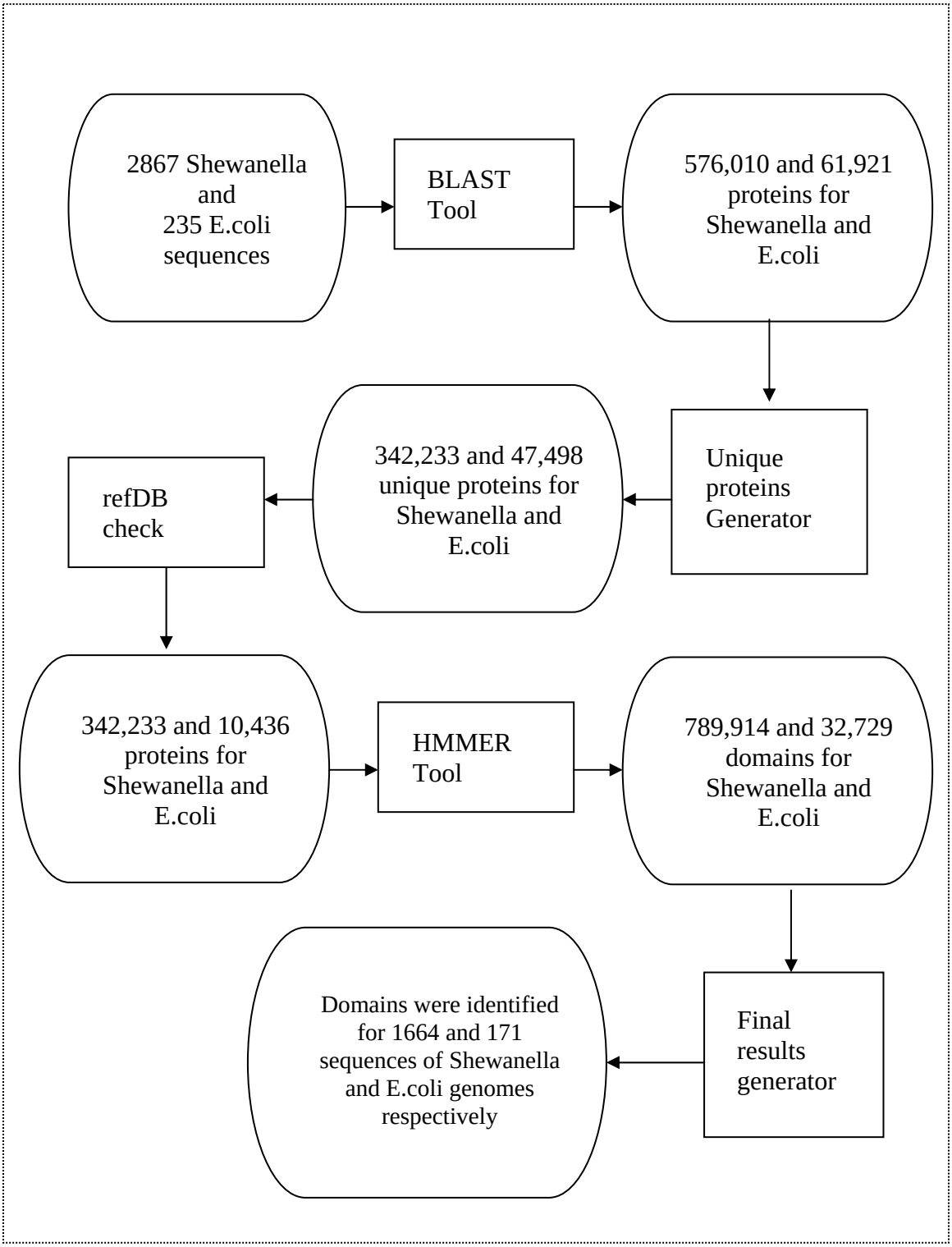


Figure 4.3: Domain Identification Automated Tool chain results flow

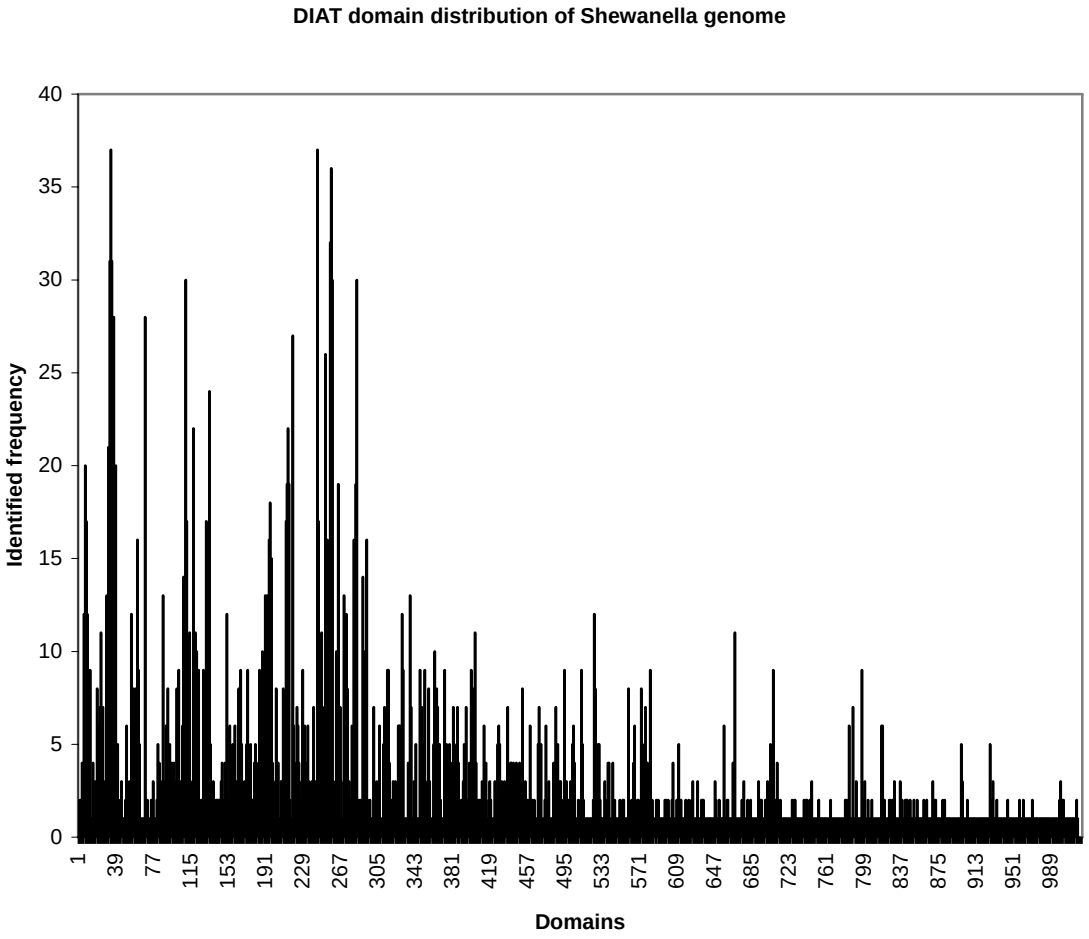


Figure 4.4: Resulted DIAT domain distribution of Shewanella genome

For some peptides more than one domain was found. For 171 unknown regions of the E.coli genome, a total of 420 domains were identified including 248 unique domains and their distribution is shown in Figure 4.5. For some unknown regions more than one domain was found.

Tables 4.1 and 4.2 show domains that were identified by DIAT tool chain in unknown regions of the Shewanella and E.coli genomes respectively. Table 4.1 shows all the domain models that were identified by DIAT tool chain in 20 or more unknown regions of the Shewanella genome. For E.coli genome top 20 identified domain models were retrieved based on the frequency of occurrence in unknown regions as shown in Table 4.2. For example from Tables 4.1 and 4.2 it is clear that the PAS domain models were identified in unknown regions of both Shewanella and E.coli genomes, hence these domain models need to be modified so that HMMER identifies these unknown regions. This statistics of the identified domains will help us with genome wide domain modeling of the Shewanella and E.coli genomes.

4.3 DVAT Results

All the unknown regions for which no domains are identified in the DIAT tool chain are used as input sequences for the DVAT tool chain. For the Shewanella

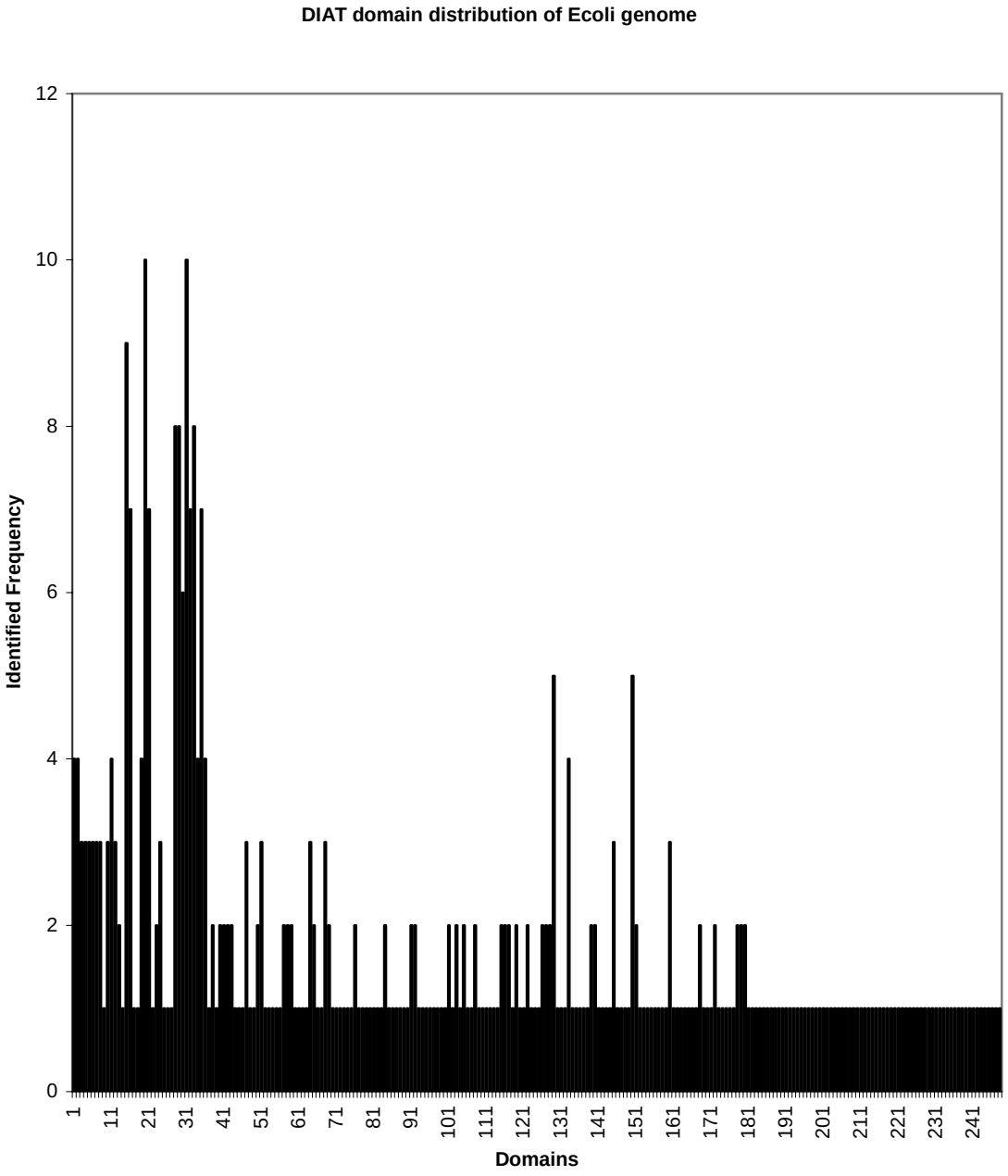


Figure 4.5: Resulted DIAT domain distribution of E.coli genome

Table 4.1: DIAT domain statistics for Shewanella genome

Domain Name	Frequency of occurrence
PD40	20
HisKA	21
TPR_1	31
TPR_2	37
TPR_4	31
TPR_3	24
Sel1	28
SBP_bac_3	20
Fer4	28
PKD	30
Big_2	22
LacI	24
Epimerase	22
Rve	27
HAMP	37
GGDEF	26
PAS_4	32
PAS	36
PAS_3	30
TonB_dep_Rec	30

Table 4.2: DIAT domain statistics for E.coli genome

Domain Name	Frequency of occurrence
Fil_haemagg	4
Pertactin	4
Big_1	4
TonB_dep_Rec	9
Plug	7
DEAD	4
Helicase_C	10
ResIII	7
HisKA	8
PAS	8
PAS_3	6
PAS_4	10
HAMP	7
GAF	8
HATPase_c	4
GGDEF	7
HisKA_3	4
SMC_N	5
ABC_tran	4
Molydop_binding	5

and E.coli genomes a total of 1203 and 64 unknown regions were submitted to the DVAT tool chain. The DVAT tool chain is similar to the DIAT tool chain except now for each query sequence we have ten input sequences as explained in chapter 3. For query sequences that resulted in ten or less PSI-BLAST protein matches in DIAT, all the matching protein regions are used as inputs. Hence the problem size and PSI-BLAST computation time increases for the DVAT tool chain when compared to the DIAT tool chain. On the other hand, as most of these input sequences are similar they resulted in many repeated proteins that reduced the problem size and *hmmpfam* computation time for the DVAT tool chain.

For the Shewanella genome, the total number of input sequences for the DVAT tool chain are 9,770. The PSI-BLAST searches for these query sequences resulted in a total of 994,960 proteins including only 100,978 unique proteins. Because 833,932 of the proteins are repeated, the search time for HMMER tool is significantly reduced for the DVAT tool chain. For the E.coli genome, 734 input query sequences resulted in 112,431 significant protein matches from PSI-BLAST searches, including only 14735 unique proteins. This classification of proteins resulted in saving significant amount of computation time. A database check is performed to verify how many of these proteins already existed in the refDB to further save HMMER computation time. For the Shewanella genome out of 100,978 unique proteins, 75,648 had information in the refDB database, so *hmmpfam* was run on only 25,330 proteins.

And for E.coli out of 14,735, *hmmpfam* was run on only 2264 proteins with the information for the rest of the proteins retrieved from the refDB database. The FINALOUT files for all these query sequences were built using the PROTOUT files and domain information from the updated refDB database. For the Shewanella and E.coli genomes, out of 1203 and 64 query unknown regions, domains were identified for 340 and 26 sequences, hence these are not submitted to the DDAT tool chain. The remaining sequences, 863 from Shewanella and 38 from E.coli, are potential sequences for domain discovery that are sent to the DDAT tool chain. Figure 4.6 illustrates the DVAT results flow.

Statistics were taken for Shewanella genome for 340 unknown regions for which domains were identified in at least one of the ten query sequences used in DVAT tool chain. A total of 1347 domains were identified, out of which 312 domains were unique and their distribution is shown in Figure 4.7. A total of 98 domains were identified for 26 unknown regions of the E.coli genome, out of which 33 domains were unique and their distribution is shown in Figure 4.8. For some peptides more than one domain was found. Tables 4.3 and 4.4 show domains that were identified by DVAT tool chain in unknown regions of the Shewanella and E.coli genomes respectively. From Tables 4.1, 4.2, 4.3, and 4.4 it is clear some of these domain models in Pfam database are not complete to detect all unknown regions that match those domains. Hence with the use of DVAT tool chain Pfam domain models can be modified even considering the distant relatives. For example ‘rve’ domain from Table 4.3 is identified by DVAT tool

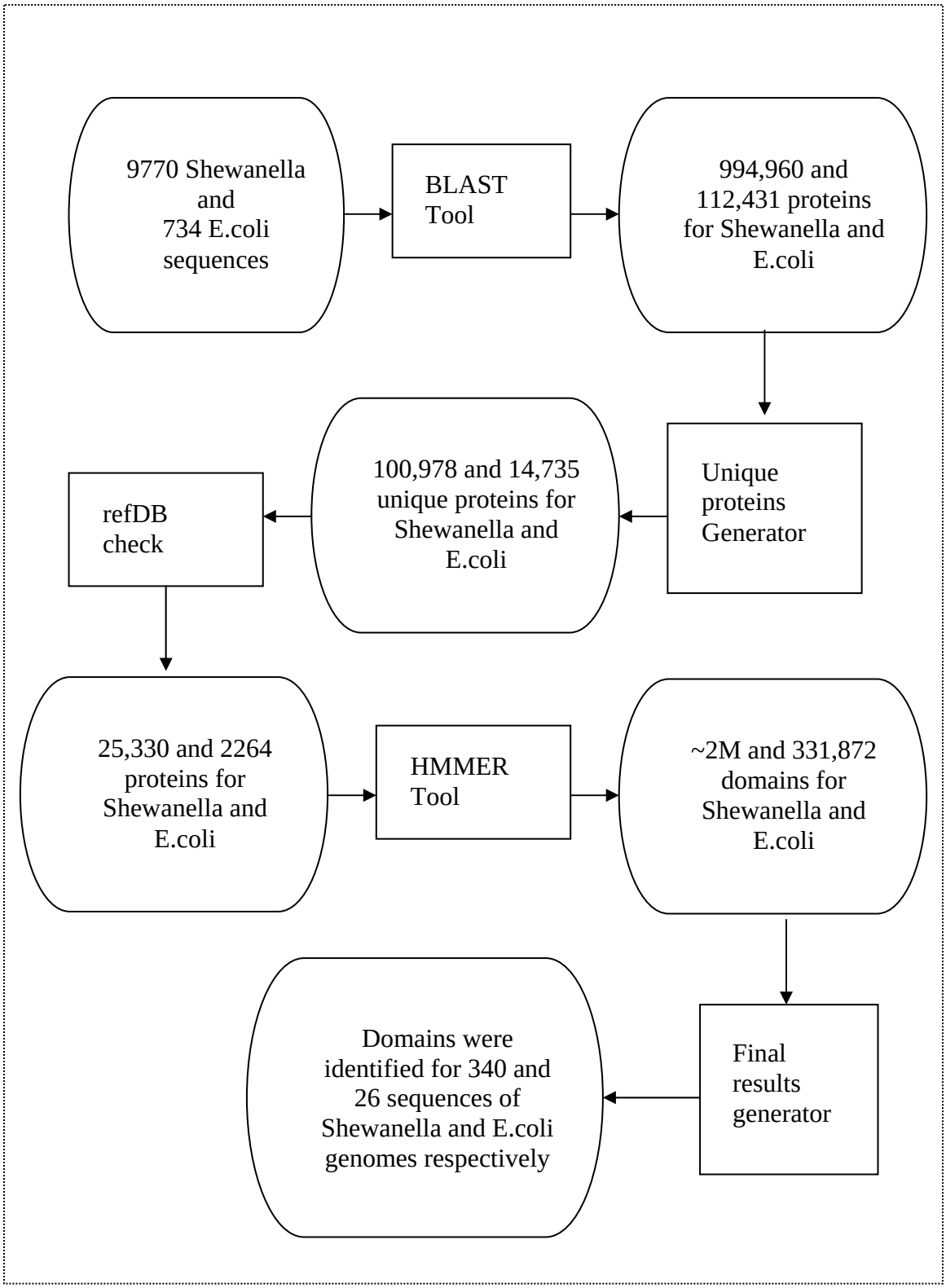


Figure 4.6: Domain Verification Automated Tool chain results flow

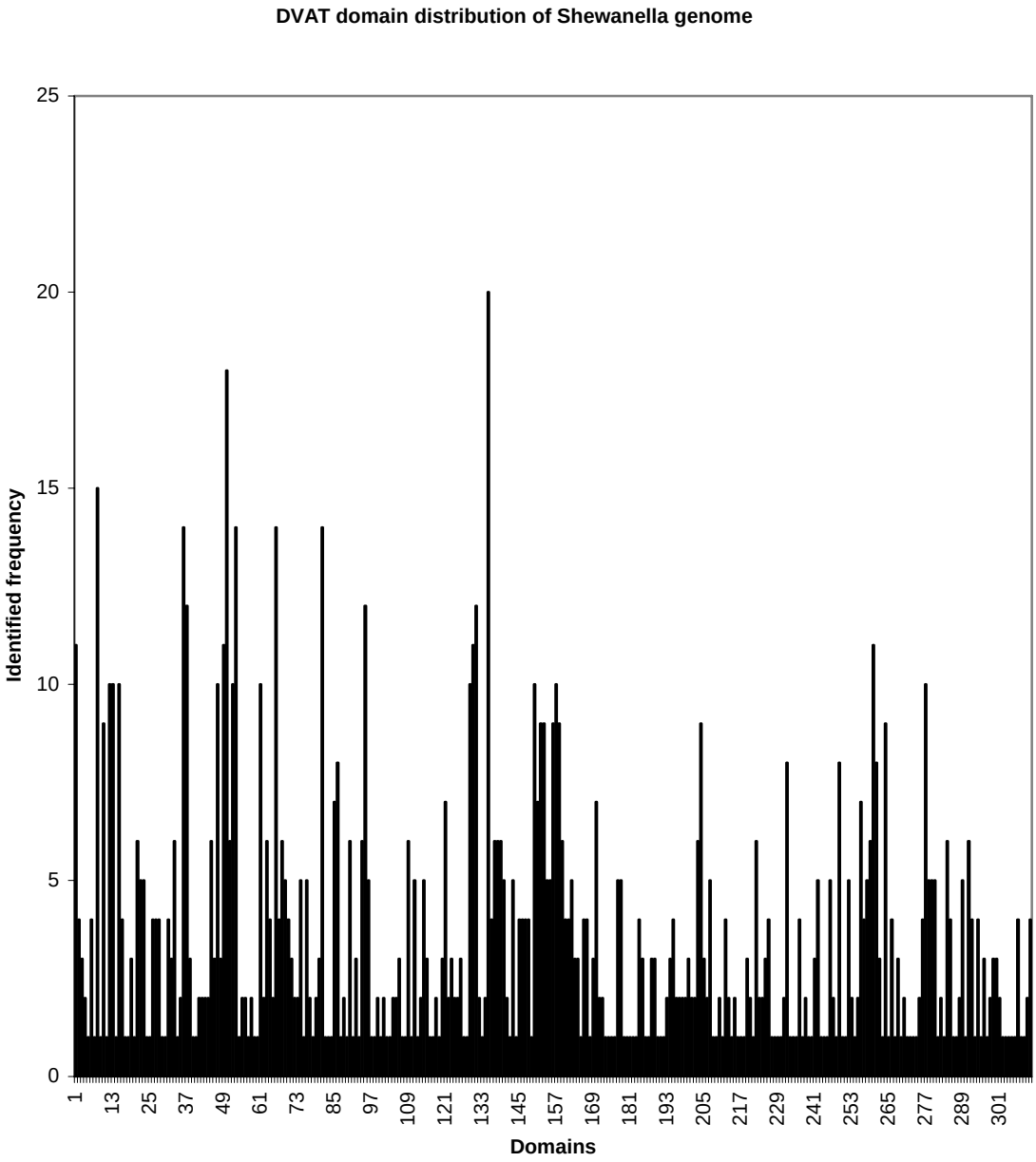


Figure 4.7: Resulted DVAT domain distribution of Shewanella genome

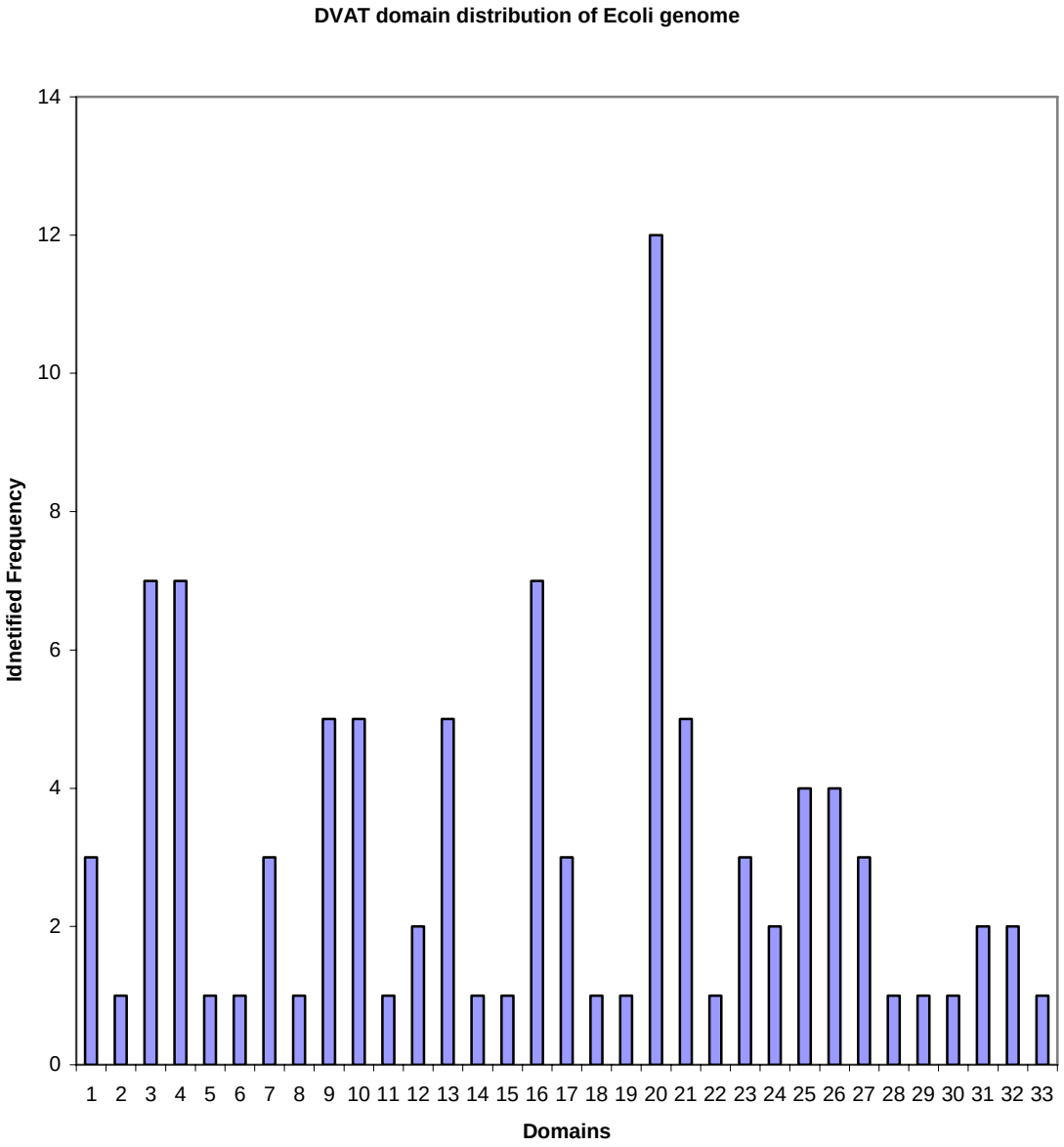


Figure 4.8: Resulted DVAT domain distribution of E.coli genome

Table 4.3: DVAT domain statistics for Shewanella genome

Domain Name	Frequency of occurrence
MMR_HSR1	11
AT_hook	15
MTS	10
Methyltransf_12	10
Methyltransf_11	10
rve	257
PAS_3	14
Abhydrolase_1	12
HisKA	10
GAF	11
HAMP	18
PAS	10
PAS_4	14
Involucrin	10
PT	14
ABC_tran	14
Molybdopterin	12
GXGXG	10
DnaJ_CXXCXGXG	11
Cytochrom_C552	12
Helicase_C	20
UPF0020	10
CMAS	10
Radical_SAM	11
YfaZ	10

Table 4.4: DVAT domain statistics for E.coli genome

Domain Name	Frequency of occurrence
Exonuc_VII_L	7
tRNA_anti	7
DHH	5
CCG	5
DnaG_DnaB_bind	5
Sigma70_r4_2	7
Molydop_binding	12
Molybdopterin	5

chain in 257 different unknown regions that were missed by HMMER search using this domain model. Thus ‘rve’ domain model need to be revised to identify these missing sequences.

4.4 DDAT Results

All the unknown regions for which no domains are identified in either the DIAT or the DVAT tool chains are used as input sequences to the DDAT tool chain. For the Shewanella and E.coli genomes, a total of 863 and 38 unknown regions were inputted to the DDAT tool chain. The PSI-BLAST protein matches for these sequences are retrieved from the PROTOUT files and the matching regions of these similar proteins are used to construct the multiple sequence alignments. The multiple sequence alignments are used to build the HMMs. These new domain models are used to search against the nr database using *hmmsearch* and the matching proteins are retrieved from the nr database. A final check is performed to retrieve known domains as explained in section 3.3. All the sequences for which no known domains are identified are plausible regions for new domains.

For the Shewanella genome DDAT is run on 863 unknown regions. The HMMs are built and these domain models are used to search against the nr database. The resulting proteins from the search are matched to the protein list generated

by the DIAT and DVAT tools. The search resulted in 22,690 unique proteins of these only 16,420 had no information in refDB. For these 16,420 proteins *hmmpfam* was run and the results were stored in refDB. For the E.coli genome DDAT is run on 38 unknown regions. The HMMs are built and searched against the nr database resulting in 4201 proteins that were not found in PROTOUTs of the DIAT and DVAT tools, of these proteins only 989 had no information in refDB. For these 989 proteins *hmmpfam* was run and the results were stored in refDB.

FINALOUT files are constructed comparing the matching regions of the protein matches resulted from the *hmmsearch* and the domains identified for these regions by *hmmpfam*. A final check is performed to see whether there are any known domains in these proteins matching regions. For the Shewanella genome, 13 out of the 863 unknown regions have known domains in the similar protein regions. None were found for the E.coli genome. So a total of 850 new domain models were discovered for the Shewanella genome and 38 new domain models for the E.coli genome. These new domain models will be further investigated individually and could be added to Pfam database.

Figure 4.9 shows the pictorial representation of the genome-wide protein domain modeling of Shewanella using the automated tool chains.

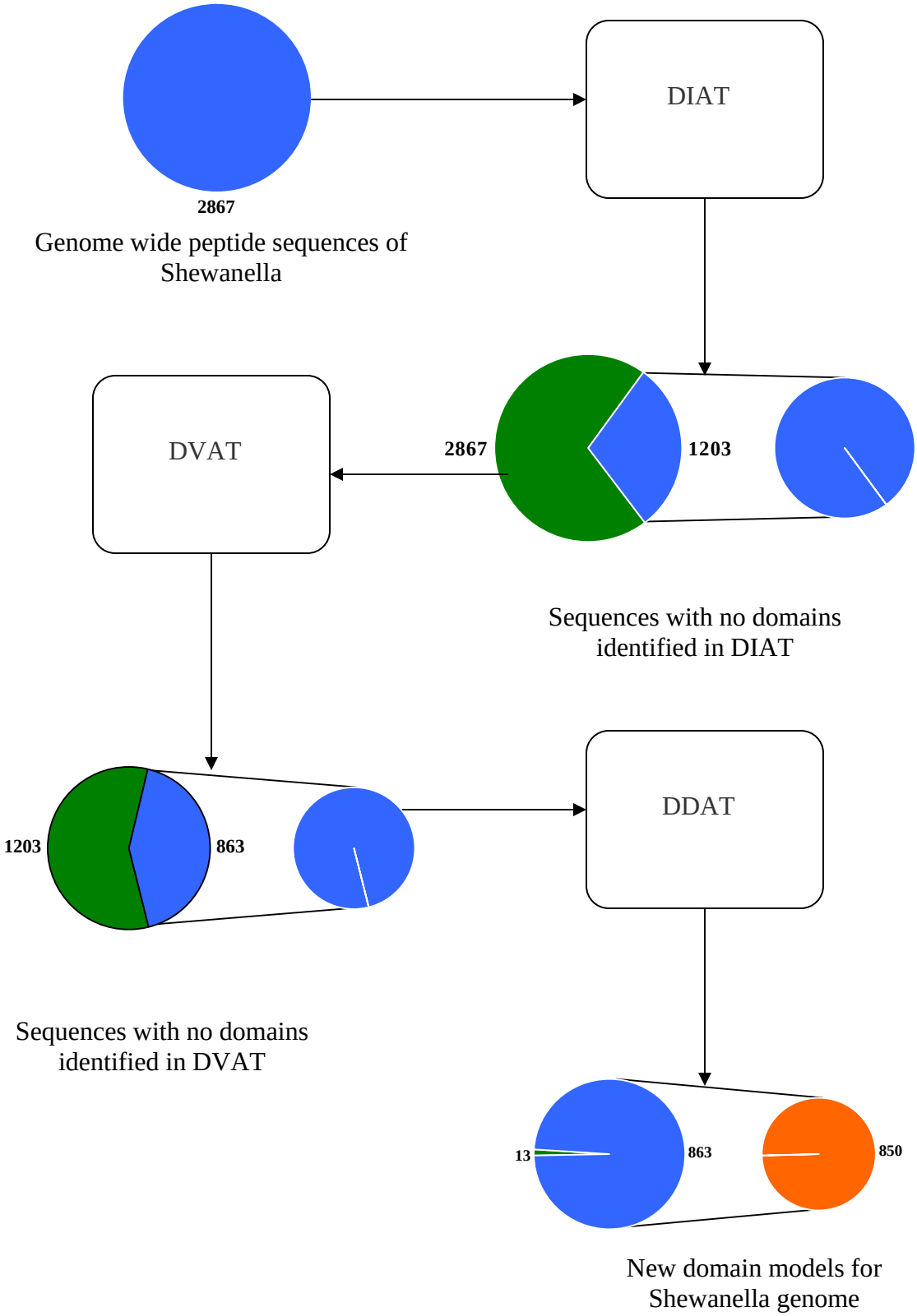


Figure 4.9: Automated tool chain flow using Shewanella genome

4.5 Domain Model Verification Results

Using PHYRE [117] search secondary structures of some newly discovered domain models are retrieved. The new domain models were matched from 100% to 0% to the existing secondary structures. The domains that matched 100% will have the properties of the matched domains. The domains with 0% match are completely new domains for which no knowledge is found in the current databases. A total of 20 models for Shewanella and 30 models for E.coli were constructed using PHYRE. Out of these 50 models only 15 models had estimated precision match greater than 50% with the know domain models or proteins. This shows the percentage of novel domain models yet to be discovered is higher than already existing models, which indicates a huge scope of discovering new domain knowledge. This shows the robustness and efficiency of the automated tool chain in discovering new knowledge. Figure 4.10 and 4.11 shows secondary structure matches of one Shewanella and one E.coli domain models generated by PHYRE search.

The next chapter describes the computation times for running the automated tools, a study of performance metrics for better resource utilization, and some job distribution techniques.



Figure 4.10: Shewanella domain model with 100% precision with EF-G C-terminal domain from PHYRE search. The red areas indicate alpha helixes, blue areas indicate beta sheets, and gray areas indicate coil regions.

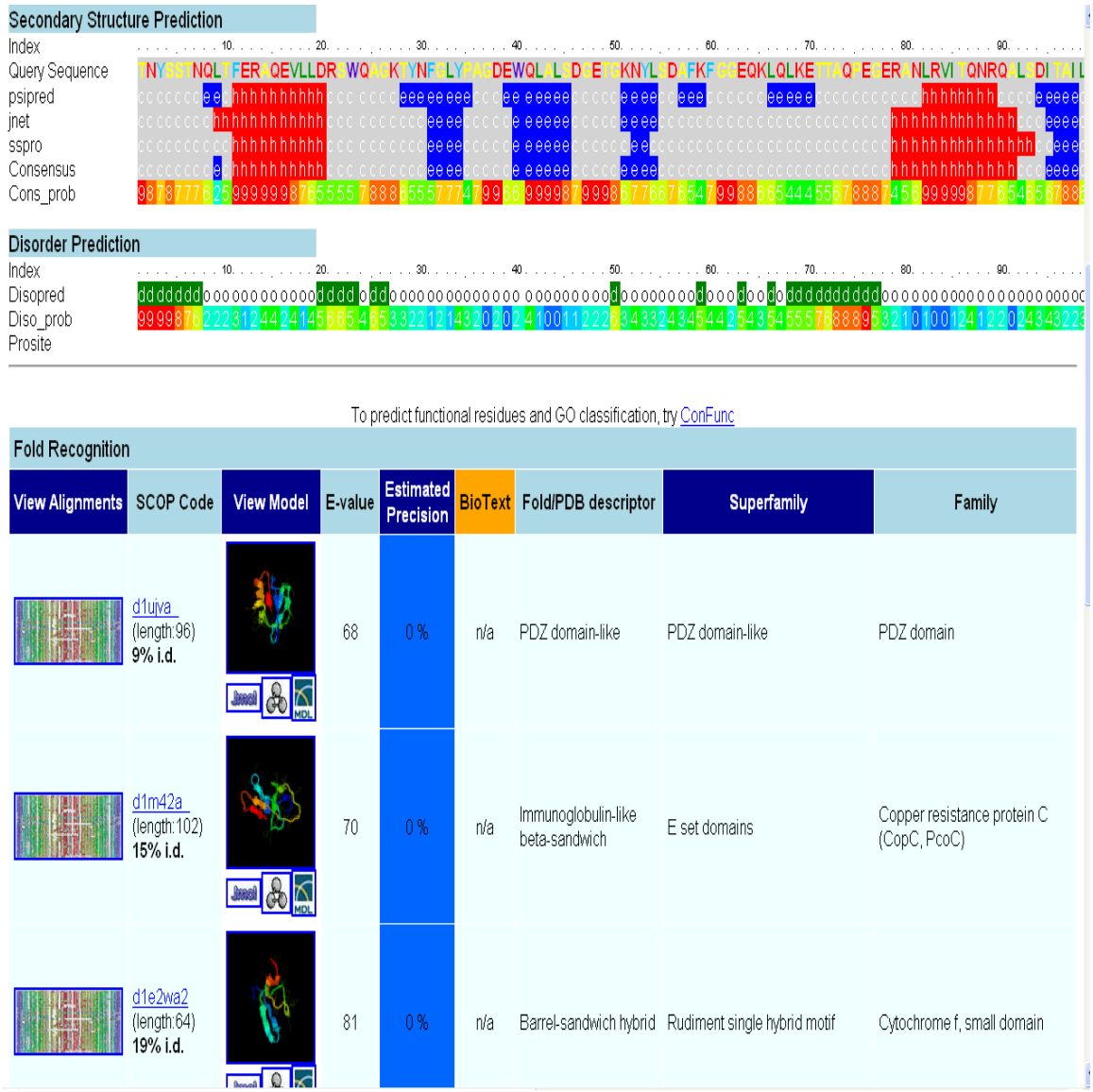


Figure 4.11: E.coli domain model with 0% precision with PDZ domain from PHYRE search. The red areas indicate alpha helixes, blue areas indicate beta sheets, and gray areas indicate coil regions.

Chapter Five

Computational Results

One has to perform thousands of PSI-BLAST searches, tens of thousands of HMMER searches, and hundreds of multiple sequence alignments per genome for domain modeling. There are hundreds of sequenced genomes with millions yet to be sequenced. Thus scaling and performance evaluation play a major role in speeding up this process, efficiently using the resources, and managing the results. This chapter discusses the computation time required to model Shewanella and E.coli genomes. This chapter also investigates architectural assessments of different processor architecture such as Opteron, Sparc, and Xeon, along with multicore and threading assessments. Finally concludes with the job mapping and distribution algorithms for cluster computing.

The characteristics of BLAST and HMMER algorithms are shown in Table 5.1. The MUSCLE tool space complexity is $O(N^2 + L^2)$, and time complexity is $O(N^4 + NL^2)$ where *L* is the typical sequence length and *N* is the number of sequences. MUSCLE takes comparatively less time to execute. MUSCLE is the fastest multiple sequence alignment tool, faster than the most commonly used CLUSTALW [99]. Most of the computation time is spent for *hmmpfam* searches of HMMER tool when compared to any other tool used by the automated tool chains. Thus this chapter focuses more on characterizing the *hmmpfam* searches.

Table: 5.1: BLAST and HMMER suite statistics.

BLAST suite	HMMER suite
Algorithms	
blastpgp	*hmmpfam*
Computational Complexities	
O(MN) where M is length query sequence, N is the number of protein sequences in database	O(QT) where Q is the length of the query sequence, T is the number of Domain models. O(mQT) m is number of proteins
Databases	
nr protein database (Jan 2007)	Pfam_ls (PFAM21 domain database)
Database size	
~3 GB with around 3 million proteins	~700MB with around 9000 families
Operations	
Integer intensive	Integer intensive
Multi-threaded	
Yes	Yes
Output file types	
TXT and XML	TXT

5.1 Architectural Assessment

The various test bench architectures used to run the PSIBLAST, and HMMER tools are Sun 1350 MHz SparcV9 dual core processors with 4GB RAM, Dell 3.20GHz Intel(R) Xeon(TM) dual core processors with 4GB RAM, and AMD 2.60GHz Opteron(tm) Y255 dual core processors with 4GB RAM. Various protein sequences of lengths varying from 100 amino acids to 20,000 amino acids are used for generating computation times on all these architectures. The computation times for PSI-BLAST and *hmmpfam* on these three architectures are shown in Figures 5.1 and 5.2 respectively.

From the Figures 5.1 and 5.2 it is clear that the Opteron out performed the Xeon and Sparc, and Xeon had a better performance than Sparc. We had access to Sun Sparc clusters and Intel Xeon cluster. The choice is obvious we picked a cluster of Intel Xeon processors for generating the protein domain modeling results. The cluster has 12 nodes and each node have two dual core 3.20GHz Intel Xeon processors with 4GB RAM. Hence the cluster had a total of 24 dual core processors. This cluster is used to generate protein domain models for Shewanella and E.coli genomes.

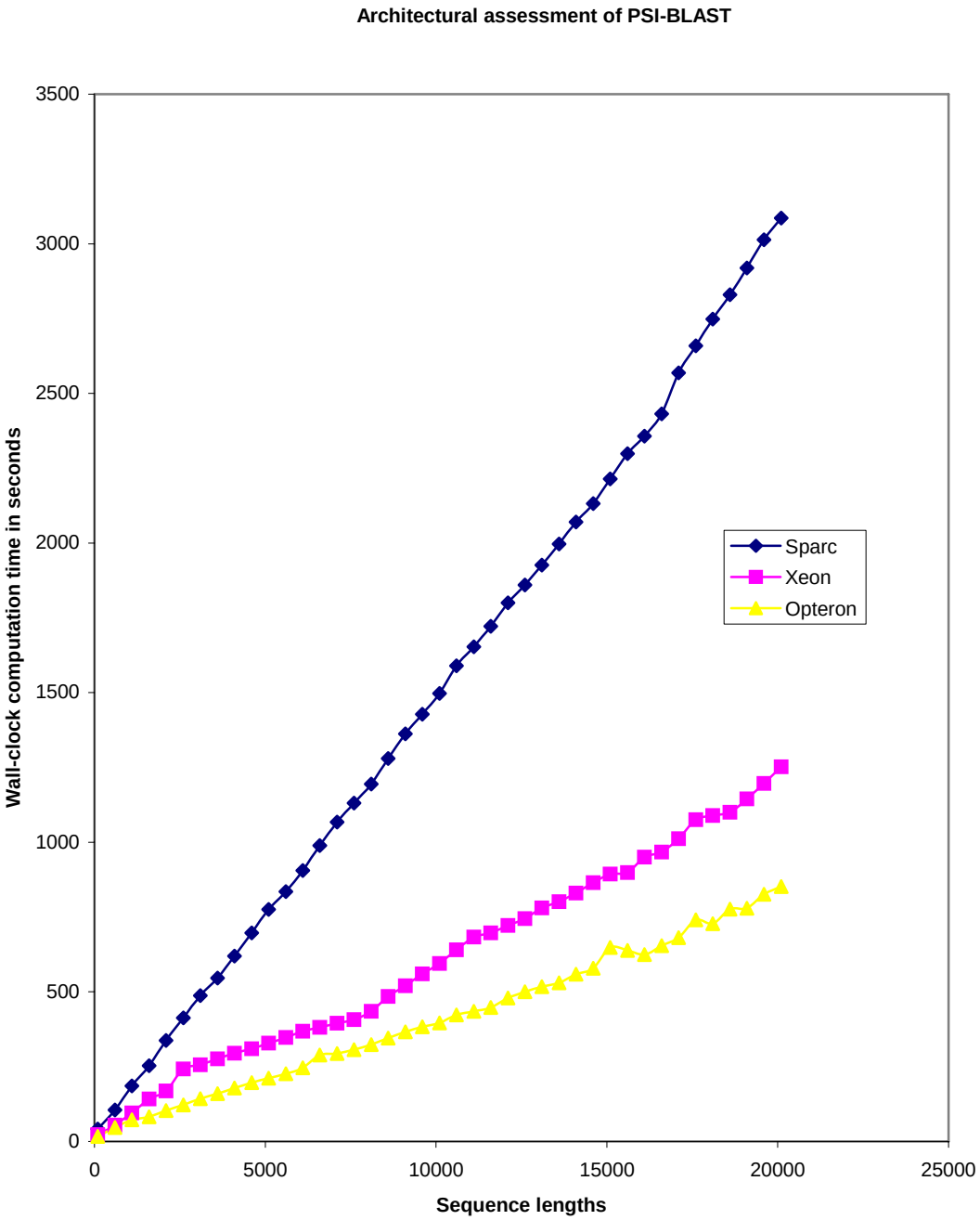


Figure: 5.1: Comparison plot of PSI-BLAST computation times between Sun Sparc, Intel Xeon, and AMD Opteron.

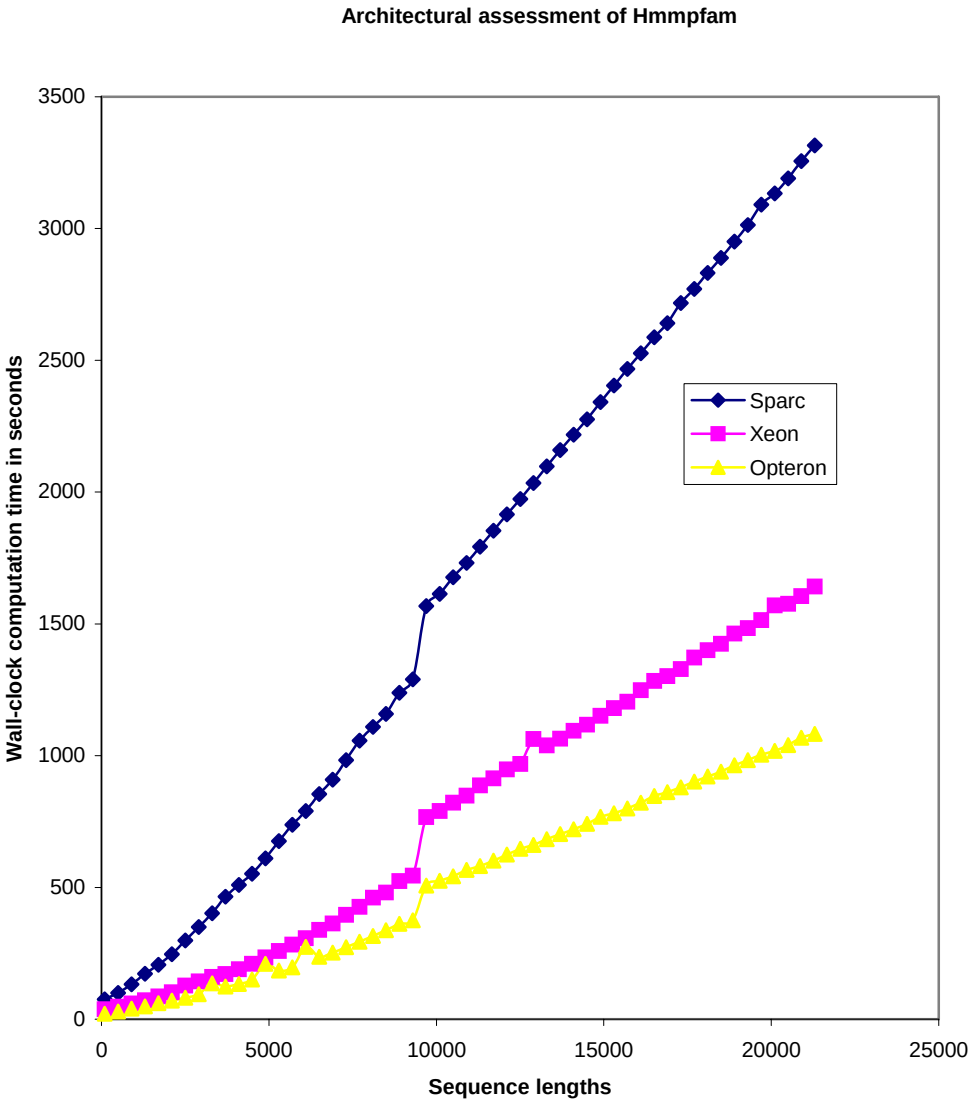


Figure: 5.2: Comparison plot of HMMER computation times between Sun Sparc, Intel Xeon, and AMD Opteron.

5.2 Computation Times for Shewanella and E.coli Genome-Wide Domain Modeling

The MANGEN and GENWIDEshew files have a total of 100 and 2867 input unknown regions. The MANGEN file is used to validate the DIAT tool that is described in the section 5.4. The computation time for both test bench files for the Shewanella genome using DIAT tool chain is shown in Table 5.3.

These are initial time measurements recorded for running the DIAT tool chain on the Shewanella genome. There are no initial refDB entries, so refDB was built using the results from first DIAT run on GENWIDEshew file. Later this refDB was used to generate DOMOUT files and for the proteins with no domain information in refDB, *hmmpfam* was used to get domain information and the refDB is updated. This saved significant amount of time for DVAT and DDAT tool chain runs. The total time taken to run DVAT and DDAT on Shewanella is shown in the Table 5.4. Both DVAT and DDAT tool chains were run using same cluster as DIAT. The significant decrease in the computation time using refDB will be clearer from the Figures 5.3 and 5.4.

The GENWIDEecoli test bench file has only 235 input unknown regions resulting in smaller computation times. The PSIBLAST of DIAT and DVAT tool chain for GENWIDEecoli test bench file resulted in 47498 and 14735 proteins respectively. Only 30% of DIAT resultant proteins and 15% of DVAT resultant proteins did not

Table 5.3: Comparison of DIAT on GENWIDEshew and MANGEN files

Test bench files	DIAT
MANGEN	~20 hrs
GENWIDEshew	~9 days

Table 5.4: Computation times of DVAT and DDAT for GENWIDEshew file

Tools	GENWIDEshew
DVAT	~95 hrs
DDAT	~74 hrs

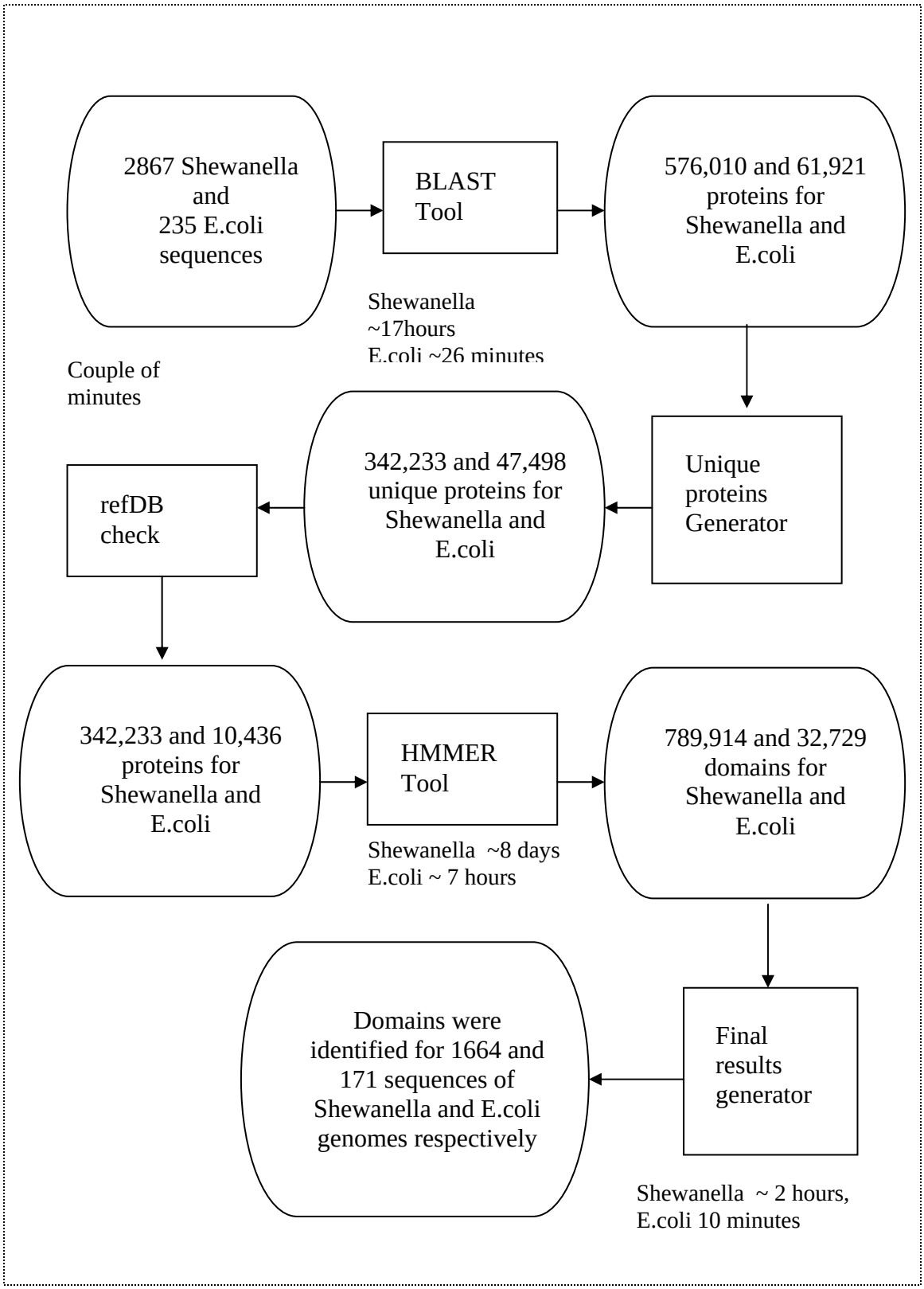


Figure 5.3: Domain Identification Automated Tool chain computation time results flow

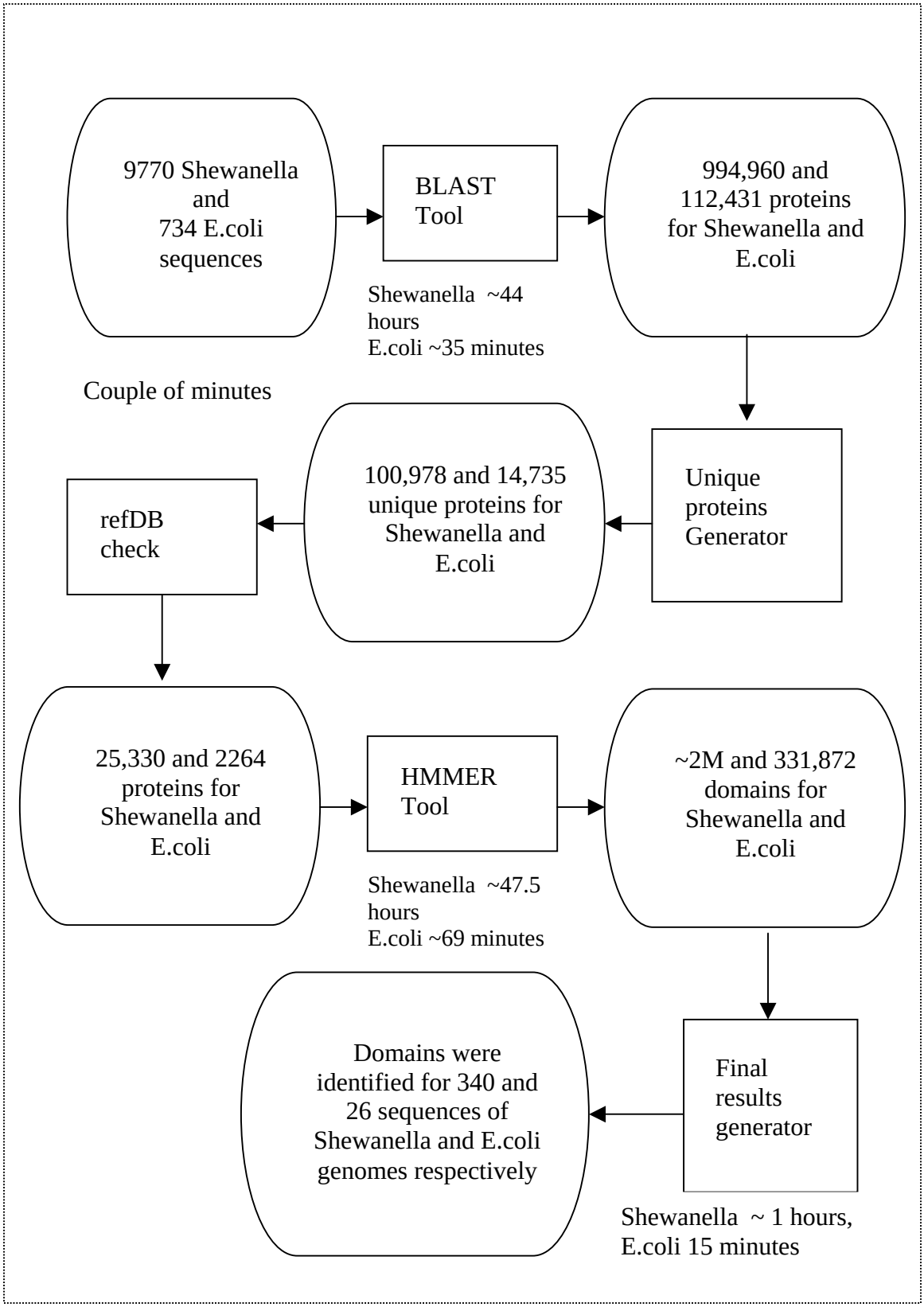


Figure 5.4: Domain Verification Automated Tool chain computation time results flow

have any information in refDB database. Hence hmmpfam was run only on these proteins thus saving significant amount of time. The Table 5.5 shows the computation times for DIAT, DVAT, and DDAT tools for GENWIDEecoli test bench file. The reduction in the computation time from days to hours is because of the intelligent use of the existing domain information from refDB.

5.2 Multicore Architectures and Threading

The popularity of multicore processors is increasing, as their performance is better than single core processors. The multicore processor has more than one CPU (Central Processing Unit) on the chip and respective caches. An Intel dual core processor has two CPUs and both the CPUs share a single coherent cache on the other hand AMD dual cores have individual caches. This is the reason why AMD Opteron outperformed Intel Xeon in the previous section. The goal of this study is to utilize the multicore architecture in the clusters for optimum work distribution.

Table 5.5: Computation times of DIAT, DVAT, and DDAT for GENWIDEecoli file

Tools	GENWIDEecoli
DIAT	~8 hrs
DVAT	~3 hrs
DDAT	~3 hrs

The threading capability of both the PSI-BLAST and HMMER tools are explored to test number of core utilization versus computation time. The HMMER tool allocates two threads by default and PSI-BLAST tool allocate only one thread by default. Since PSI-BLAST and HMMER are the two most extensively used tools in the automated tool chains, studies were conducted to optimize resource usage for the faster completion of these tasks. A dual-core (2 CPUs) Intel Xeon processors is picked for the study. One other study was performed using Intel Xeon single core processor on which hyper-threading was enabled. The operating system sees two processors instead of one if hyper-threading technology is enabled.

The threading functionality of both BLAST and HMMER tools are explored to derive optimized number of threads for a query sequence of particular length. Three sets of data are collected for *blastpgp* and *hmmpfam* runs. The first set of data is generated using only one of the two cores of the Xeon processor, always with a load on the second core. The second set of data is generated using both cores of the processor. And the single core Xeon processor generates the third set.

Neither PSIBLAST nor HMMER uses the dual core functionality of the processor to cut down the computation time to half. The performance obtained using threads is not greatly affected by the dual core architecture even though using one core of dual core processor outperformed single core architecture. One more

interesting discovery, as the sequence length increases the number of threads required for optimum performance increases. For PSIBLAST runs using one core 3-4 threads give good performance. For PSIBLAST using both cores of dual core processor runs using 3-4 threads give good performance for sequences less than 5000 amino acids and 4-5 threads for sequences greater than 5000 amino acids. For best performance of time and cost, PSIBLAST should be run on one core using 4 threads.

HMMER does not take advantage of threading for sequence lengths smaller than 200-300 amino acids using one core of the dual core processor. For sequences of length 400 or above 3-4 threads gives good performance. Using both cores of the dual core processor for sequences smaller than 500 AAs 2 threads gives good performance. For best performance of time and cost, HMMER should be run on one core using 3 threads.

5.3 Validation of DIAT

First, two PSI-BLAST iterations are run in the DIAT tool chain, *hmmpfam* is run on the proteins resulted from the PSI-BLAST searches for domain identification. Out of 97 unknown regions for 70 different proteins in the MANGEN file, the DIAT tool chain identified domains for 61 unknown regions; DIAT missed only 3 protein hits. The reason for this is the domains for a protein were identified in iteration 5

Table 5.6: Statistics for Domains identified for MANGEN file of Shewanella genome using DIAT.

Test bench file	Unknown regions with domain hits	Missed domains by DIAT
MANGEN (2-iterations)	61	3
MANGEN (4-iterations)	70	0

and the standard practice is to perform only 4 iterations. The other two proteins domains were identified in iteration 3 and DIAT performed only 2 iterations. DIAT identified domains for 16 proteins that were not found manually, as DIAT used the latest Pfam database (PFAM21). Next the DIAT tool chain was run with four PSI-BLAST iterations, this time there were no missed protein hits. The DIAT tool chain identified domains for all the proteins similar to manual search along with hits for an additional 25 proteins. The results in Table 5.6 show the robustness of the tool chain and its contribution to new knowledge.

5.4 Solved Programming Challenges

The automated tool chain was designed using the PHP (Hypertext Preprocessor) scripting language. PHP is a server-side HTML (Hypertext Markup Language) embedded scripting language. PHP is helpful to make the results web accessible very easily and it is simple to parse the resultant files in different formats to populate various databases. The results form PROTOUT and DOMOUT files are

used to populate the refDB database. The refDB database was generated using SQLite and also a text file of the entire refDB database was generated. The refDB was used to generate the domain information and from which the final resultant files are derived. Two programs were generated one to retrieve information from the SQLite database and another from the text file database. The program written to retrieve the data from text file used hashing technique and was three times faster than retrieving data from SQLite database. Hence the text file refDB was used to retrieve the domain information. Hashing technique was also used to compare PROTOUT and DOMOUT files to generate FINALOUT files faster thus saving lot of parsing time.

5.5 Job Mapping and Distribution

The problem size is huge; currently we have 12.9 million proteins in our database. The distribution of sequence lengths of these 12.9 million proteins is shown in the Figure 5.5. Statistics were derived from protein length distributions that shows 98.2% of all proteins have lengths less than 1000 amino acids. The average protein length is 269 amino acids with a standard deviation of 265 amino acids. The minimum and maximum lengths are 2 and 36,800 amino acids with 75 % of all proteins sequences have lengths less than 320 amino acids. Hence we need an efficient job-mapping algorithm for optimum performance to evenly distribute these proteins of varying lengths across computing clusters that is the primary goal of this research.

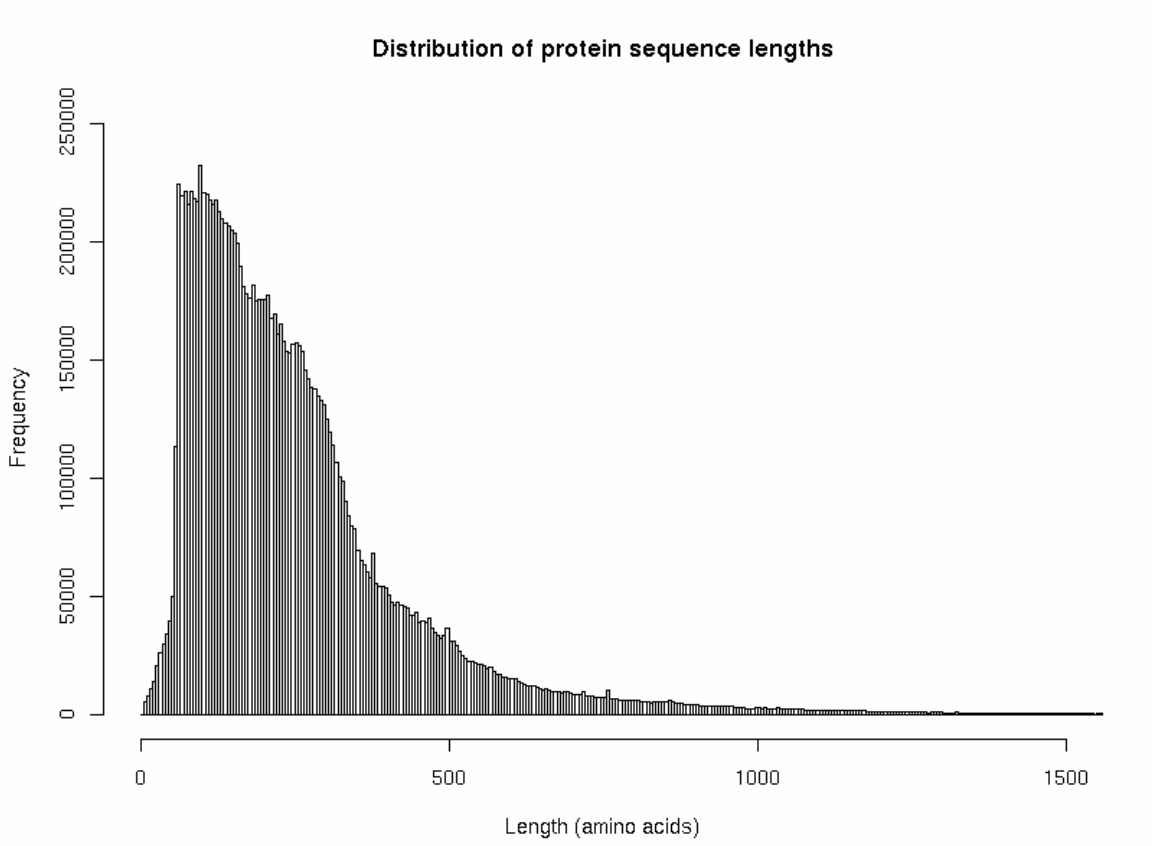


Figure: 5.5: The distribution of protein sequence lengths of 12.9 million protein sequences currently available (Image added with the permission of Luke Ulrich).

Two different approaches for scheduling the jobs on the cluster are discussed in this section. One is the algorithmic approach for PSI-BLAST jobs and the other is based on equations or mathematical approach for HMMER jobs. The goal of the job scheduling on the clusters is to finish the computation on all nodes at the same time so that there is no wait for one processor to finish. So based on the unknown region length and the number of sequences, each node is allocated with certain amount of work so that all nodes finish at the same time.

5.5.1 PSI-BLAST Job Scheduler

The PSI-BLAST run times are random and do not depend on amino acid lengths or number of sequences in a file. Based on number of relatives present for a proteins sequence and the number of iterations to reach convergence varies thus varying the computation time. Which means some protein sequences converge in 2 iterations and some can run for 10 iterations and still not converge. One more interesting discovery was running individual PSI-BLAST was faster than combining sequences together in a file. This led to the development of the algorithmic approach to distribute the PSI-BLAST runs individually using bins.

There are ‘P’ processing nodes in the cluster and the number of bins is ‘B’. One bin is allocated to each processing node. The jobs are distributed across the cluster using the following algorithm.

Job allocation algorithm steps for PSI-BLAST

1. Generate B=P bins
2. Populate an array with the input sequences
3. Sort the array in descending order based on sequence lengths
4. Traverse the array one by one by allotting each sequence to a bin in the order

```
binCounter=0;
foreach( array as sequence){
        binCounter++;
        Allot sequence to Bin( binCounter);
        if(binCounter==B){
                binCounter=0;
        }
}
```

Once the jobs are distributed across the bins, each bin is allocated to a processor. Next all the jobs are put into queue for each processor with biggest jobs first. The smallest sequence length of job is 80 amino acids as this was the cutoff used for input query sequences. A jobCheck program was created to identify unfinished or failed jobs and allot these failed jobs to the processor

queues with least number of jobs left. This program also checks for the empty queues and allocates the unfinished jobs for other queues to the empty queue thus keeping the balance. Using the job allocation algorithm and the jobCheck program all the processors finish the execution almost at the same time.

5.5.2 HMMER Job Scheduler

The hmmpfam run times using 16 different proteins is shown in Figure 5.6. These 16 proteins used are around 25000 AAs in length. The run times are recorded by varying the lengths of these sequences by 200 AAs starting from 80 AAs. From the Figure 5.6 it is clear that there are two linear regimes, connected by a smooth transition curve between these two regimes. The hmmpfam runs take advantage of number of sequences in a file that means the time taken to run N sequences in a file is less than the sum of time taken to run N sequences individually. Now we added one more dimension to our curve that is number of sequences in a file. We plotted a curve using three variables, length of protein sequences, number of sequences in a file, and the time taken for the runs. Figure 5.7 show the three-dimensional curve plotted using GNUPLOT. Logarithmic scale is used to plot this curve. We see the same characteristics in this plot similar to two-dimensional plot in Figure 5.6. There are two linear regimes connected by a smooth transitional region.

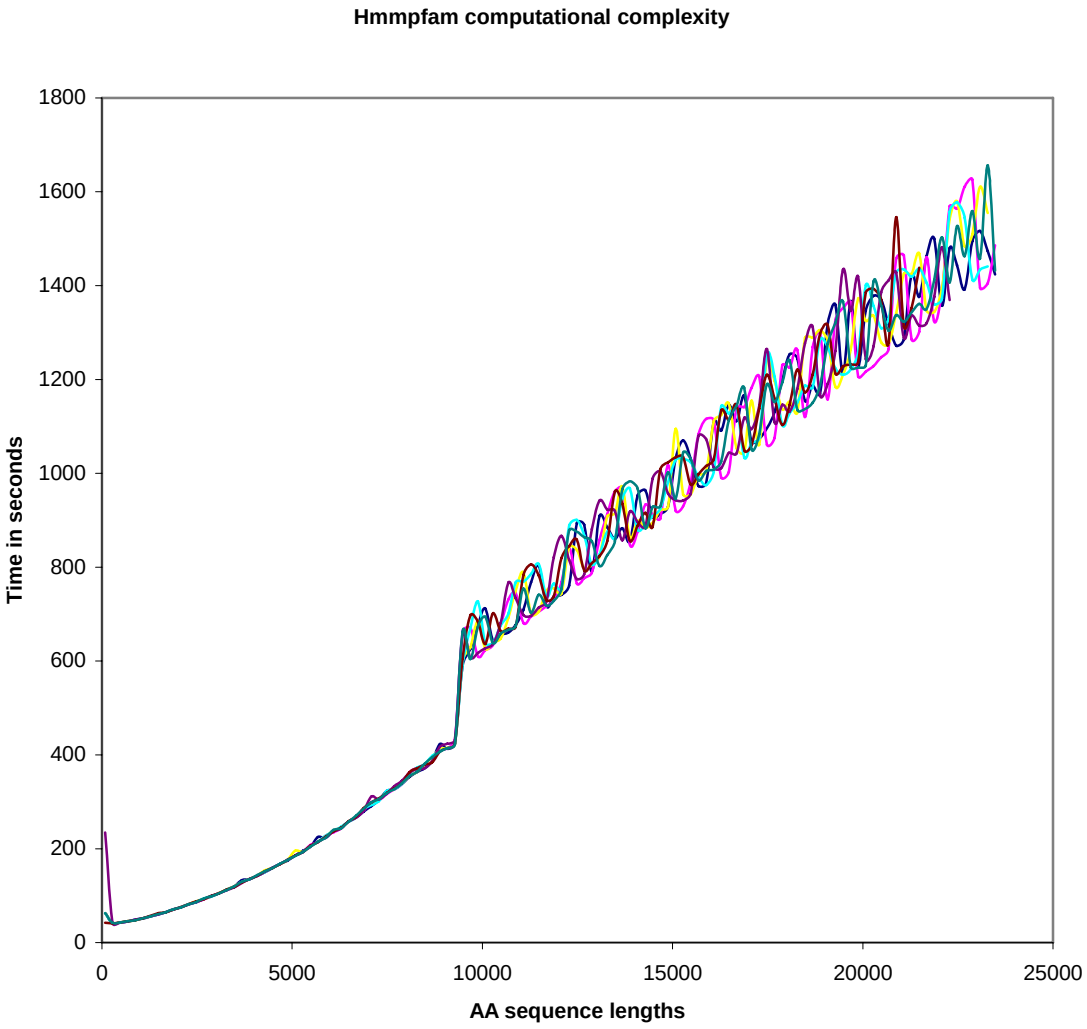


Figure: 5.6: Comparison plot between hmmpfam computation times and amino acid lengths for 16 different protein sequences of varying lengths from 100 amino acids to 24000 amino acids.

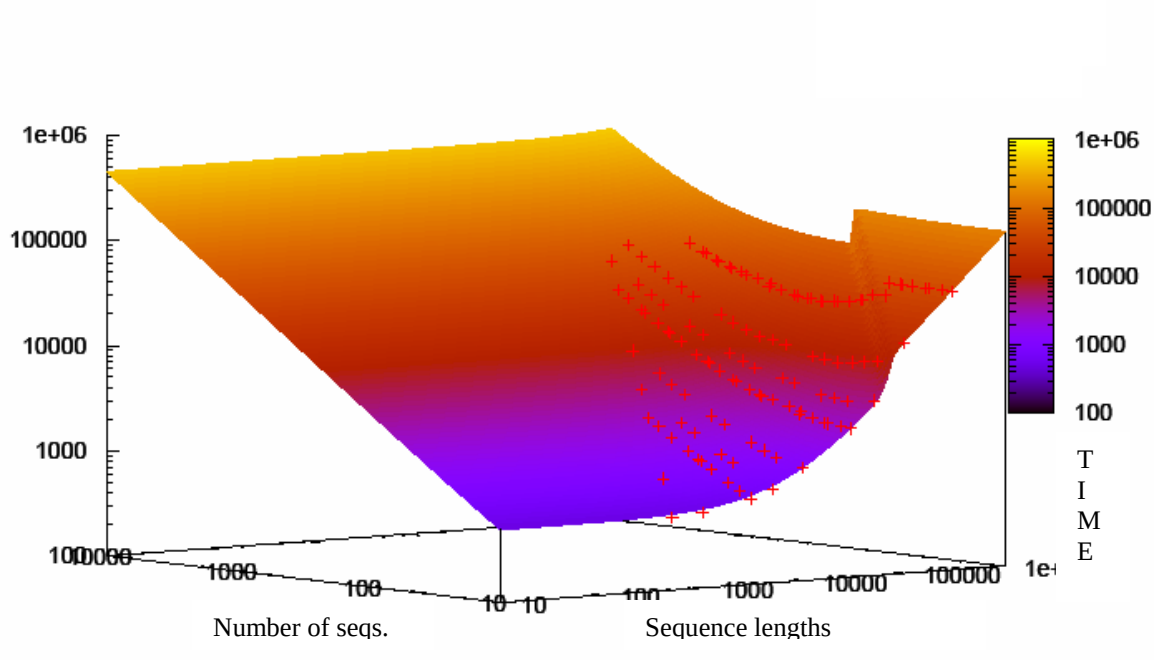


Figure: 5.7: Three dimensional comparison plot between protein sequence lengths, number of sequences in a file and their respective computation times for hmmpfam jobs

We used the least square fitting algorithm to generate a surface that connects all the points to derive equation to define our model. There are two regimes and the transition takes place ~8000 amino acids length for this architecture. We derived a surface formula that represents the time predicted to run hmmpfam on a cluster for jobs of varying lengths and number of sequences. Here AA is the total number of amino acids of all the sequences in the job and N is the number of sequences in the job. T is the time taken for the job to finish.

The *a*'s and *b*'s coefficients tells information about the computational power of the architectures on which the *hmmpfam* is run. We applied inverse tangent curve to connect these two linear regimes. The Time T that define entire model is as follows

$$T(AA,N)=\left(\frac{\left(1-\tan^{-1}\left(\frac{(AA-(c_1N+c_0))s}{1.57}\right)\right)}{2}\right)\times(a_xAA+a_0+a_yN)+\left(\frac{\left(1+\tan^{-1}\left(\frac{(AA-(c_1N+c_0))s}{1.57}\right)\right)}{2}\right)\times(b_xAA+b_0+b_yN)$$

-- equation 1

The coefficients $c_{1,0}$ are related to the edges between these two regimes. The equation (c_1N+c_0) when evaluated for *N=1* shows the maximum number of amino-acid in one sequence that it would run the program in a efficient way, in our case this number is 7800 AA. Also there is the coefficient '*s*' that defines how smooth the transition between the two regimes is. Therefore the function $\tan^{-1}$ is related to the connection between these two regimes. The two linear regions are

Table 5.7: Protein statistics in three different ranges.

Ranges	Sequence %	AA %	Computation time (hours)
Range1 (<150 AAs)	32.9	12.1	61277
Range2 (>=150 <=1500)	66.5	82.8	163303
Range3 (>1500)	0.6	5.1	4767

an efficient regime that means the error generated by the equation is negligible and the non-linear region of the curve is inefficient regime as the error in this region is higher. The jobs are distributed in the cluster considering this inefficiency.

Based on the sequence lengths the jobs are classified into three different groups. Jobs with sequence lengths less than 150(range1), the jobs in between 150 and 1500(range2), and the jobs with sequence lengths greater than 1500(range3). These numbers are picked considering the inefficiency of the equations and the statistics of sequence lengths. Out of 12.9 million sequences the percentages of number of sequences and number of amino acids, and estimated computation times in hours using equation 1 for each range is shown in the Table 5.7.

To efficiently distribute the work and to easily identify the failed or unfinished jobs, the sequences were divided into bins. The computation time for each bin is multiple of 2 hours. A program is designed to distribute the sequences in each range into bins of four hours each. The total numbers of 4-hour bins for range1, range2, and range3 are 15338, 40708, and 1145 respectively. Sample bins were

taken from each range and tested on the cluster. The estimated time and the computation time for some random sample sizes and the sample sizes of 4-hour bins and 8-hour bins are shown in Table 5.8. The samples from range 2 and range 1 are more efficient than range 3, which means the estimated time is close to the computation time. Hence the bins are allocated to the job queue in the order of range3, range1, and range2 respectively so that all the compute nodes finish the computation almost at the same time.

The next chapter put forth the achievements and contributions of this research along with conclusions and future work.

Table 5.8: The estimated and computed times for some sample files.

SeqLength_number	Computed time (seconds)	Estimated time (seconds)
61_292	14633	14423
106_274	14580	14401
124_268	14472	14425
151_259	14436	14432
172_252	14372	14414
272_225	14309	14451
875_135	13971	14403
987_126	13961	14438
1411_100	13948	14453
1645_90	13945	14498
3236_53	14851	14558
4974_36	16329	14559
7241_15	11373	15093
7718_13	10799	15924
16953_6	15329	15924
1342_207	27990	28829

Chapter Six

Conclusions and Future work

This dissertation enables genome-wide protein domain modeling, one of the most important problems in biology, using high throughput and high-resolution automation techniques with better quality, speed, and cost effectiveness than manual procedures.

This dissertation describes a new automated tool chain for protein domain modeling. This new bioinformatics application generates protein domain models much faster, which enables biologist to use their valuable time in the labs rather in front of computers. With the use of cluster computing, genome-wide protein domain modeling is made easier. With the help of supercomputing the protein domain modeling can address entire protein databases. The rate at which new protein domain knowledge can now be generated will revolutionize the science and encourage the use and design of automated bioinformatics tools.

During the course of design of the automated tool chains for protein domain modeling, many other tools were generated that are beyond the scope of this dissertation. This dissertation lead to important contributions such as:

1. Protein domain modeling automated tool chain design

   Three automated tool chains are developed, for protein domain identification (DIAT), verification (DVAT), and discovery (DDAT). These tool chains are new additions to bioinformatics tools and will be made publicly available. The knowledge from the domain discovery model will help in detecting speculative regions for possible new domain discovery.

2. Feedback on the effectiveness of Pfam HMMs

   The domain hit/miss statistics from domain identification models and domain verification models will assist us in evaluating the effectiveness of Pfam HMMs. This will assist us in suggesting the modifications for changing the HMMs based on the statistics generated for the various domains in genome-wide analysis, so that no domains are missed based on the evolutionary relatives.

3. Architectural and algorithmic assessment of automated tool chain

   Performance evaluation of multicore architectures, and clusters of computers is explored in this dissertation for better-automated job allocation. Threading functionality of BLAST and HMMER tools is explored to reduce the computation time.

4. Job mapping, task management and results management

Since there are millions of sequences to model, an effective job-mapping algorithm is designed to distribute the jobs evenly across available nodes in a cluster. A task manager script is designed to check on unfinished jobs or blocked jobs, and these unfinished jobs will be allotted to other processors so that valuable information is not lost. Finally, all results generated by the automated tool chains are uploaded into the PepDomDB database for storage and future reference.

5. Protein domain database generation

   A PepDomDB database for peptides and their respective domain information is generated. This database is ready to be public once the domain information for all genomes is populated. One major challenge is to keep this database schema scalable and up to date as new domains are discovered. The design of this database will lead to faster domain modeling of newly sequenced genomes.

6. New protein domain knowledge generation

   New protein domain models are generated for Shewanella and E.coli genomes for the peptides for which no domains are currently present in the MiST database. The statistics of various domains identified for Shewanella and E.coli genomes are documented for genome wide domain modeling. The work is in progress to generate the protein domain knowledge for all the genomes in MiST database.

This dissertation resulted in new knowledge about protein domain modeling. This dissertation also generated statistics for missed domain models using existing Pfam database to give feedback to improve domain models. This efficient domain modeling on a genome-wide scale will help biologists to solve problems like protein functions, structures and folding. The design of an automated tool chain will be greatly helpful for biologists who now perform sequence similarity analysis manually, thus saving tremendous effort that can be directed towards laboratory experimentation. The primary contribution of this dissertation is a set of automated tool chains for protein domain modeling to explore the problem of genome-wide analysis, including a good foundation for using the most appropriate architectures for huge problem sizes.

From the results it is clear that the DIAT tool chain identified all the domains that were manually generated along with some new domains for the query unknown regions, thus demonstrating its robustness and effective design. The time taken to generate the results was a few hours using cluster computing when compared to months of work done manually for 100 sequences. Using a small cluster of computers, domain models were generated for thousands of unknown regions of the Shewanella genome in few days, and hundreds of unknown regions of the E.coli genome in few hours. This showed the tool chains ability to help in the process of deriving important biologically relevant information from completely sequenced genomes.

Web access to automated tool chains for protein domain modeling needs to be designed and implemented on a dedicated cluster for public use. Publicly available PepDomDB database should be created in such a way that registered users can upload new domains and peptides after the verification process is completed. Implementing the automated tool chains on supercomputing architecture to solve bigger problems remains to be explored. Work is in progress to design an automatic job-mapping algorithm for different architectures. Finally, secondary structure predicting tools could be added to the automated tool chain for further analysis.

References

References

1. Altschul, SF, W Gish, W Miller, EW Myers, and DJ Lipman. Basic local alignment search tool. Journal of Molecular Biology 215(3):403-10, 1990
2. Altschul, SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, and DJ Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acid research, 1997,Vol 25, No 17, 3389-3402.
3. Ross, PE. The making of 24 Billion gene machine. Forbes, February 2000,21:98-104.
4. Friend, SH. How DNA microarrays and expression profiling will effect clinical practice. British Medical Journal, 319:1-2, 1999.
5. Draghici, S. Data analysis tools for DNA microarrays. 2003.
6. D. Eisenberg, E. M. Marcotte, L. Xenarios, and T. O. Yeates. Protein function in the post-genomic era. Nature, 405:823-826, 2000.
7. Mount, DW. Bioinformatics: Sequence and Genome analysis. Second edition 2004.
8. Pearson, WR. Comparison of methods for searching protein sequence databases. Protein science 4:1150-1160, 1995.
9. Gibbs, AJ, McIntyre, GA. The diagram, a method for comparing sequences. It use with amino acid and nucleotide sequences. European Journal of Biochemistry 16:1-11, 1970.
10. Smith, TF, and Waterman, MS. Identification of common molecular subsequences. J. of Molecular Biology 147:195-97, 1981
11. Needleman, SB, and Wunch, CD. A general method applicable to the search for similarities in amino acid sequences of two proteins. J. of Molecular Biology 48:443-453, 1970.
12. Pearson, WR, and Lipman, DJ. Improved tools for biological sequence comparison. Proc, Natl. Acad. Sci. 85:2444-2448, 1988.
13. Richardson, JS. The anatomy and taxonomy of protein structure. Adv Protein Chem, 34:167-339, 1981.
14. Bork, P. Shuffled domains in extracellular proteins. FEBS Lett, 286:47-54, 1991
15. Wetlaufer, DB. Nucleation, rapid folding, and globular intrachain regions in proteins. Proc Natl Acad Sci U S A, 70:697-701, 1973.
16. Savageau, MA. Proteins of Escherichia coli come in sizes that are multiples of 14 kDa: domain concepts and evolutionary implications. Proc Natl Acad Sci U S A, 83:1198-1202, 1986.
17. Jones, S, Stewart, M, Michie, A, Swindells, MB, Orengo, C, and Thornton, JM. Domain assignment for protein structures using a consensus approach: characterization and analysis. Protein Sci, 7:233-242, 1998.
18. Siddiqui, AS, and Barton, GJ. Continuous and discontinuous domains - an algorithm for the automatic generation of reliable protein domain definitions. Protein Sci, 4:872-884, 1995.

19. Pedretti, KT, Casavant, TL, Braun, RC, Scheetz, TE, Birkett, CL, and Roberts CA. Three complementary approaches to parallelization of local BLAST Service on Workstation clusters. PacT-99, LNCS 1662, 271-82, 1999.
20. Braun, RC, Pedretti, KT, Casavant, TL, Scheetz, TE, Birkett, CL, and Roberts CA. Parallelization of local BLAST Service on Workstation clusters. Future generation computer systems 17, 745-754, 2001.
21. Bjornson, RD, Sherman, AH, SB Weston, Willard,N, and Wing, J. TurboBLAST : A Parallel Implementation of BLAST Built on the TurboHub. TurboGenomics, Inc.
22. Hong-Soog Kim, Hae-Jin Kim, and Dong-Soo Han. Hyper-Blast: A parallelized BLAST on cluster system. ICCS, LNCS 2659, 213-222, 2003.
23. Darling, AE, Carey,L, and Wu-chun Feng. The design , Implementation and Evaluation of mpiBLAST.
24. Lin, H, Ma, X, Chandramohan, P, Geist, A, Samatova, Nagiza. Efficient Data access for Parallel BLAST. IPDPS, Volume 01,page 72.2, 2005.
25. Konishi, F and Konagaya A. The architectural design of high throughput BLAST services on OBIGrid. LSGRID 2004, LNBI 3370, pp.32-42, 2005.
26. Oehmen, C, and Nieplocha, J. ScalaBLAST: A scalable implementation of BLAST for high-performance data-intensive bioinformatics analysis.
27. Moore, G. Cramming more components onto integrated circuits. Electronics Magazine 1965.
28. Eddy, S. HMMer User's Guide. 1998.
29. George, RA, and Heringa, Jaap. Protein domain identification and Improved similarity searching using PSI-BLAST. Proteins: Structure, Function and Genetics 48:672-681, 2002.
30. Pagni, M, Ioannidis, V, Cerutti, l, Zahn-Zabal, M, Jongeneel, CV, and Falquet L. MyHits: a new interactive resource for protein annotation and domain identification. Nucleic Acids Research, Vol. 32, 332-335, 2004.
31. Dickens, NJ and Ponting CP. ThoR:a tool for domain discovery and curation of multiple alignments. Genome Biology, 2003, 4:R52.
32. Schultz, J, Milpetz, F, Bork, P, and Pointing, CP. SMART, a simple modular architecture research tool: Identification of signaling domains. Proc. Natl. Acad. Sci., Vol. 95, 5897-5864, 1998.
33. Schultz, J, Copley, RR, Doerks, T, Pointing, CP and Bork, P. SMART: a web-based tool for the study of genetically mobile domains.
34. Coin, L, Bateman, A, and Durbin, R. Enhanced protein domain discovery using taxonomy. BMC Bioinformatics, 5:56, 2004.
35. Cheng, J, Sweredoski, MJ, Baldi, P. DOMpro: protein domain prediction using profiles, secondary structure, relative solvent accessibility and recursive neural networks.
36. Eddy, SR. Profile hidden Markov models. Bioinformatics, 14:755-763, 1998.

37. Krogh, A, Brown, M, Mian, IS, Sjolande, K, and Haussler D. Hidden Markov models in computational biology. Application to protein modeling. Journal of Molecular Biology 235:1501-31, 1994
38. Hughey, R, and Krogh A. Hidden Markov models for sequence analysis: extension and analysis of basic method. Comput. Appl. Biosci. 12:95-107, 1996.
39. Landman, J, Ray, J, Walters, JP. Accelerating HMMer searches on Opteron processors with minimally invasive recoding. IEEE AINA, 2006.
40. Costa, RLC, and Lifschitz, S. Database allocation strategies for parallel BLAST evaluation on clusters. Distributed and Parallel Databases, 13, 99-127, 2003.
41. Braun, RC, Pedretti, KT, Casavant, TL, Scheetz, TE, Birkett, CL and Roberts, CA. Parallelization of local BLAST service on workstation clusters. Future Generation Computer Systems, 17, 745-754, 2001.
42. Rangwala, H, Lantz, E, Musselman, R, Pinnow, K, Smith, B and Wallenfelt, B. Massive parallel BLAST for the Blue Gene/L.
43. Tan, G, Xu,L, Feng, S, and Sun, N. An experimental study of optimizing Bioinformatics Applications. IEEE, 2006.
44. Kang, JY, Gupta, S, and Gaudiot, JL. An efficient PIM (Processor-In-Memory) architecture for BLAST. IEEE, 2004.
45. Lancaster, JM. Design and evaluation of a BLAST ungapped extension accelerator. Master's thesis, Washington University, Saint Louis, 2006.
46. Srinivasan, U, Che, PS, Diao, Q, Lim CC, Li, E, Chen, Y, Ju, R, and Zhang Y. Characterization and analysis of HMMER and SVM-RFE parallel bioinformatics applications. IEEE 2005.
47. Wun, B, Buhler, J, and Crowley, P. Exploiting coarse-grained parallelism to accelerate protein motif finding with a network processor. Parallel Architecure and compilation Techniques, 173-184, IEEE, 2005.
48. Horn, DR, Houston, M, and Hanrahan, Pat. ClawHMMER: A streaming HMMer-Search implementation. Proc. IEEE Supercomputing 2005.
49. Maddimsetty, RP, Buhler J, Chamberlian, RD, Franklin, MA, and Harris, B. Accelerator design for protein sequence HMM search. ICS, 2006.
50. Landman, J, Ray, J, and Walters, JP. Accelerating HMMer searches on Opteron processors with minimally invasive recoding. Proc. Or the 20th International Conference on Advanced Information Networking and Applications, IEEE, 2006.
51. Ulrich, L, and Jouline, IB. MiST: a microbial signal transduction database. Nucleic Acids Research, Vol. 35, 386-390, 2007.
52. Park, J, Karplus, K, Barrett, C, Hughey, R, Haussler D,Hubbard, T, and Chothia, C. Sequence comparisons using multiple sequence detect three times as many remote homologues as pairwise methods. Journal of Molecular Biology, 184:1201-1210, 1998.
53. SQLite database speed comparison. http://www.sqlite.org/speed.html
54. Initial sequencing and analysis of human genome. Nature Vol. 409, 860-921, 2001

55. Ross, PE. The making of a 24 billion gene machine. Forbes, February 21:98-104, 2000.
56. Venter, JC, Smith, HO, and Hood, l. A new strategy for genome sequencing. Nature 381(6581):364-6, 1996.
57. Venter, JC, Adams, MD, Sutton, GG, Kerlavage, AR, Smith, HO. And Hunkapiller M. Shotgun sequencing of the human genome. Science, Vol. 280, no. 5369, 1540-1542, 1998.
58. National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov/
59. http://bioweb.pasteur.fr/seqanal/blast/#psiblast
60. PSI-BLAST tutorials. http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-2.html
61. Pieper, U, Eswar, N, Davis FP, Braberg H, Madhusudhan, MS, Rossi, A, Marti-Renom, M, Karchin R, Webb, BM, Eramian, D, Shen, MY, Kelly, L, Melo, F, and Sali, A. MODBASE: a database of annotated comparitive protein structure models and associated resources. Nucleic Acids Research, Vol.34, database issue, 291-295, 2006.
62. Mulder, NJ, Fleischmann W., and Apweiler R. InterPro as a new tool for whole genome analysis. A comparative analysis of Mycobacterium tuberculosis, Bacillus substilis and Echerichia coli as a case study. Regulation and Structure. Vol. 2, 35-37, 2000.
63. Hulo, N, Bairoch, A, Bulliard, V, Cerutti, L, Castro, ED, Langendijk-Genevaux, PS, Pagni, M, and Sigrist, CJA. The PROSITE database. Nucleic Acids Research. Database issue 34, 227-230, 2006.
64. Bateman A, Birney E, Cerruti L, Durbin R, Etwiller l, Eddy SR, Griffiths-Jones S, Howe KL, Marshall M, Sonnhammer ELL. The Pfam protein families database. Nucleic acids Research, 30:276-280, 2002.
65. Attwood TK, Bradley P, Flower DR, Gaulton A, Maudling N, Mitchell AL, Mpulton G, Nordle A, Paine K, Taylor P et al. . PRINTS and its automatic supplement, prePRINTS. Nucleic acids Research, 31:400-402, 2003.
66. Corpet F, Servant F, Gouzy J, Kahn D. ProDom and ProDom-CG: tools for protein domain analysis and whole genome comparisons. Nucleic acids Research 28:267-69, 2000.
67. Marchler-Bauer, A, et al. CDD: a conserved domain database for interactive domain family analysis. Nucleic acids Research 35, 237-240, 2007.
68. Henikoff S, and Henikiff J, G. Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. 89:10915-10919, 1992.
69. Dayhoff, MO. Survey of new data and computer methods of analysis. In Atlas of protein sequence and structure, vo. 5, suppl. 3. National Biomedical Research Foundation, Georgetown University, Washington, D.C. 1978.
70. Wilbur WJ, and Lipman, DJ. Rapid similarity searches of nucleic acid and protein data banks. Proc. Natl. Acad. Sci. 80:726-730, 1983

71. Bairoch, A., and Apweiler, R. The SWISS-PROT protein sequence data bank and its supplement TrEMBL. Nucl. Acids Res. 24, 21-25, 1995.
72. Kulikova T., Akhtar R., Aldebert P., Althorpe N., Andersson M., Baldwin A., Bates K., Bhattacharyya S., Bower L., Browne P., Castro M., Cochrane G., Duggan K., Eberhardt R., Faruque N., Hoad G., Kanz C., Lee C., Leinonen R., Lin Q., Lombard V., Lopez R., Lorenc D., McWilliam H., Mukherjee G., Nardone F., Garcia-Pastor M.P., Plaister S., Sobhany S., Stoehr P., Vaughan R., Wu D., Zhu W., Apweiler R. EMBL Nucleotide Sequence Database in 2006. Nucleic Acids Research 35: D16-D20, 2007.
73. Tateno, Y, and Gojobori, T. DNA Databank of Japan in the age of information biology. Nucl. Acids Res. 24, 14-17, 1996.
74. Gavin Sherlock, et al. “The Stanford Microarray Database”, *Nucleic Acids Research*, 29(1). 2001.
75. Schena,M., Shalon,D., Davis,R.W. and Brown,P.O. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, 270, 467–470. 1995.
76. Pollack JR, et al, Genome-wide analysis of DNA copy-number changes using cDNA microarrays. *Nature Genet.*, 23, 41–46. 1999.
77. Rekapalli, B, Peterson, G, Rose, J and Hillhouse B. Parallel algorithm for genetic KNN-impute algorithm. Parallel and Distributed Computing Systems Conference, 19, 171-178, 2006.
78. Peterson JD, Umayam LA, Dickinson T, Hickey EK, White O. The Comprehensive Microbial Resource. *Nucleic Acids Res.* Jan 1;29(1):123-5, 2001
79. Wheeler,D.L., Church,D.M., Edgar,R., Federhen,S., Helmberg,W., Madden,T.L., Pontius,J.U., Schuler,G.D., Schriml,L.M., Sequeira,E., *et al*. Database resources of the National Center for Biotechnology Information: update. *Nucleic Acids Res*, 32, , D39–D45, 2005.
80. Kim D. Pruitt*, Tatiana Tatusova and Donna R. Maglott. NCBI Reference Sequence (RefSeq): a curated non-redundant sequence database of genomes, transcripts and proteins. Nucleic Acids Research, 33(Database Issue):D501-D504, 2005.
81. http://www.ncbi.nlm.nih.gov/
82. Brown, TA. Genomes, Second Edition 2002.
83. Branden, C and Tooze, J. Introduction to protein structure, second edition, 1999.
84. World Wide Web
85. National Human Genome Research (NHGRI), by artist Darryl Leja.
86. Bader, J. S., Chaudhuri, A., Rothberg, J. M. & Chant, J. Gaining confidence in high-throughput protein interaction networks (2004) Nat. Biotechnol.
87. Borziak, K and Zhulin IB. FIST: a sensory domain for diverse signal transduction pathways in prokaryotes and ubiquitin signaling in eukaryotes. Bioinformatics, page 1-4, 2007.

88. Strohmaier, E, Dongarra, JJ, Meuer, HW, Simon, HD. Recent trends in the marketplace of high performance computing. Parallel Computing, volume 31, p261-273, 2005.
89. Eadline, D. Preparing for the revolution maximizing dual core technology. Basement Supercomputing, 2006.
90. Augen, J. In silico biology and clustered supercomputing shaping the future of the IT industry. Biosilico., 1, 47-49, 2003.
91. Bader, DA. Computational biology and high-performance computing. Communications of the ACM, vol. 47, no. 11, p34-41.
92. Akhurst, TJ. The role of parallel computing in bioinformatics. Thesis, 2005.
93. Zomaya, AY. Parallel Computing for bioinformatics and computational biology. 2006
94. Rekapalli, B. Genomic data analysis using grid-based computing. MS thesis, 2003.
95. Cheung,KH, Miller, P, Sherman, A, Stratmann, SWE, and Schultz, M. Graphically-enabled integration of bioinformatics tools allowing parallel execution. Journal of the American Medical Informatics Association, Suppl S, 141-145, 2000.
96. Flynn, M., Some Computer Organizations and Their Effectiveness, IEEE Trans. Comput., Vol. C-21, pp. 948, 1972.
97. Wikipedia article. http://en.wikipedia.org/wiki/Flynn%27s_Taxonomy
98. http://bioweb.pasteur.fr/seqanal/blast/
99. Thompson JD, Higgins DG, Gibson TJ. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic Acids Res* 22:4673-4680, 1994
100. Notredame C, Higgins DG, Heringa J. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol* 302(1):205-17, 2000.
101. Katoh K, Misawa K, Kuma K, and Miyata T. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res, 30:3059-3066,* 2002.
102. Simossis A and Heringa J. PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. Nucleic Acids Res. W289–W294, 2005.
103. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research* 32(5), 1792-97, 2004.
104. Subramanian AR, Weyer-Menkhoff J, Kaufmann M, and Morgenstern B. DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *Bioinformatics, 6:66,* 2005.
105. HMMER documentation. http://hmmer.janelia.org/
106. Rost B. Review: Protein secondary structure prediction continues to rise. Journal of structural biology, 2001

107. Kim H. and Park H. Protein secondary structure prediction based on an improved vector machine approach. Protein Engineering vol. 16 no. 8 pp. 553-560, 2003
108. Doong SH and Yeh CY. Secondary structure prediction using SVM and clustering. Proceedings of the Fourth International Conference on Hybrid Intelligent Systems (HIS'04) - Volume 00, P297-302, 2004
109. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. J. Mol. Biol. 292: 195-202,1999.
110. J. Cheng, A. Randall, M. Sweredoski, P. Baldi, SCRATCH: a Protein Structure and Structural Feature Prediction Server, Nucleic Acids Research, Web Server Issue vol. 33, 72-76, 2005.
111. B Rost, G Yachdav and J Liu. The PredictProtein Server. Nucleic Acids Research 32(Web Server issue):W321-W326, 2004**.**
112. Cuff, J. A., Clamp, M. E., Siddiqui, A. S., Finlay, M. and Barton, G. J. Jpred: A Consensus Secondary Structure Prediction Server, *Bioinformatics* 14:892-893, 1998.
113. Rost B. PHD: predicting one-dimensional protein structure by profile-based neural networks. Methods Enzymol, 266:525-39, 1996.
114. Ulrich LE and Zhulin IB. Four-helix bundle: a ubiquitous sensory module in prokaryotic signal transduction. Bioinformatics, Vol. 21, piii45-iii48, 2005.
115. Kimura, M. A simple model for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. Journal of Molecular Evolution 16: 111-120, 1980.
116. PROTDIST. http://evolution.genetics.washington.edu/phylip/doc/protdist.html
117. LA Kelley, RM MacCallum, MJ Sternberg. Enhanced genome annotation using structural profiles in the program 3D-PSSM. J. Mol. Biol, vol 299, pg 499-520, 2000
118. Jonathan D. Partridge, Colin Scott, Yue Tang, Robert K. Poole, and Jeffrey Green. *Escherichia coli* Transcriptome Dynamics during the Transition from Anaerobic to Aerobic Conditions. J. Biol. Chem., Vol. 281, Issue 38, 27806-27815, September 22, 2006
119. http://www.shewanella.org/whyShewanella.html
120. http://en.wikipedia.org/wiki/Image:Zinc-finger-dot-plot.png
121. Herbordt MC, Model J, Gu Y, Sukhwani B and VanCourt T. Single Pass, BLAST-Like, Approximate String Matching on FPGAs. IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM, 2006.
122. Muriki K, Underwood KD, and Sass R. RC-BLAST: towards a portable, cost-effective open source hardware implementation. Parallel and Distributed Processing Symposium, proceedings, 19th IEEE international, 2005.
123. Sotiriades E, and Dollas A. Design space exploration for the BLAST algorithm implementation. 15th annual IEEE symposium on Field-Programmable Custon Computing Machines, pp. 323-326, 2007

Vita

Bhanu Prasad Rekapalli was born in Hyderabad, a city of great historic importance in Andrapradesh, India in 1978, son of Lakshmi Sulochana, and Subba Rao Rekapalli. He attended Siva Sivani public school, which has high standards of education that laid a strong foundation for the higher studies. After performing brilliantly in a highly competitive exam, toughest of its kind in the country, he joined Jawaharlal Nehru Technological University in electrical and electronics engineering. During his undergraduate education, he got good foundation in mathematics and basic sciences that broadened his horizon of knowledge. He was an executive member of the electrical engineering department that enhanced his leadership skills. He did his practical training during the final year of undergraduate program in VLSI division of Electronics Corporation India Limited, and the research project helped him to delve deeper into challenging fields of microelectronics systems and VLSI design. He received a B.Tech. degree in Electrical and Electronics Engineering, in June 1999 from Jawaharlal Nehru Technological University.

He came to United States of America for his graduate education, he joined University of Tennessee, Knoxville. During his stay in UTK he gained experience in different kinds of jobs. He tutored mathematics to sophomores and juniors. He was also both teaching assistant and research assistant in ECE department. The teaching experience endowed him to gain proficiency in teaching.

After that he joined WebServices group of Office of Information Technology (OIT), UTK as a Graduate assistant. His job at WebServices helped him a lot with his M.S. thesis and Ph.D. dissertation. At the same time he enhanced his computer skills in database management, design and development of website applications. He became one of the UT experts in microarray analysis working as scientific programmer for UT Microarray Database. Which resulted in good publications and knowledge in the field of microarray analysis. He also designed and taught a graduate level class in “Microarray Technology and UT Microarray Database Application”. After receiving the M.S. degree in Electrical Engineering from UT, he enrolled into Ph.D. at UT.

During the course of his Ph.D. he gained knowledge in the fields of genomics and proteomics. While working in Dr. Jouline’s lab he became proficient in designing automated bioinformatics tools for proteomics applications that later became his dissertation topic. His current research interests are in the fields of bioinformatics, high performance computing, and computer architecture design along with microarray analysis. This knowledge of various fields gave him an opportunity to work as a Postdoc in Oakridge National Labs. His hobbies and interests include playing golf, working out in gym, water sports, traveling around the world and dancing.