MULTI-GENOME ANNOTATION OF GENOME FRAGMENTS USING HIDDEN MARKOV MODEL PROFILES

A THESIS SUBMITTED TO THE GRADUATE DIVISION OF THE
UNIVERSITY OF HAWAI'I IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE

DECEMBER 2007

By
Mark Menor

Thesis Committee:

Guylaine Poisson, Chairperson
Kyungim Baek
Henri Casanova

We certify that we have read this thesis and that, in our opinion, it is satisfactory in scope and quality as a thesis for the degree of Master of Science in Computer Science.

THESIS COMMITTEE

_____

Chairperson

_____

_____

Acknowledgements

I would like to dedicate this work to the memory of Will Gersch, whose teachings I will not forget. I would also like to thank my advisor, Guylaine Poisson, and Kyungim Baek for all the guidance and support that made this research possible. I am also very grateful to Yannick Gingras for his help in the automation of the full-scale viral taxonomy skeleton construction and querying system.

Abstract

To learn more about microbes and overcome the limitations of standard cultured methods, microbial communities are being studied in an uncultured state. In such metagenomic studies, genetic material is sampled from the environment and sequenced using the whole-genome shotgun sequencing technique. This results in thousands of DNA fragments that need to be identified, so that the composition and inner workings of the microbial community can begin to be understood. Those fragments are then assembled into longer portions of sequences. However the high diversity present in an environment and the often low level of genome coverage achieved by the sequencing technology result in a low number of assembled fragments (contigs) and many unassembled fragments (singletons). The identification of contigs and singletons is usually done using BLAST, which finds sequences similar to the contigs and singletons in a database. An expert may then manually read these results and determine if the function and taxonomic origins of each fragment can be determined.

In this thesis, an automated system called Anacle is developed to annotate, following a taxonomy, the unassembled fragments before the assembly process. Knowledge of what proteins can be found in each taxon is built into Anacle by clustering all known proteins of that taxon. The annotation performances from using Markov clustering (MCL) and Self-Organizing Maps (SOM) are investigated and compared. The resulting protein clusters can each be represented by a Hidden Markov Model (HMM) profile. Thus a "skeleton" of the taxon is generated with the profile HMMs providing a summary of the taxon's genetic content. The experiments show that (1) MCL is superior to SOMs in annotation and in

running time performance, (2) Anacle achieves good performance in taxonomic annotation, and (3) Anacle has the ability to generalize since it can correctly annotate fragments from genomes not present in the training dataset. These results indicate that Anacle can be very useful to metagenomics projects.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Full Name/Term	Abbreviation
Adenine	A
Artificial neural network	ANN
Base pair(s)	bp
Basic local alignment search tool	BLAST
Best-matching unit	BMU
Bioinformatics Laboratory	BiL
Blocks substitution matrices	BLOSUM
Cytosine	C
Deoxyribonucleic acid	DNA
Double stranded DNA	dsDNA
False negative	FN
False positive	FP
False positive rate	FPR
Guanine	G
Hidden Markov Model	HMM
Human herpesvirus 1	HHV-1
Information & Computer Sciences	ICS
International Committee on Taxonomy Of Viruses	ICTV
Integrated Microbial Genomes	IMG

Markov clustering	MCL
Messenger RNA	mRNA
National Center for Biotechnology Information	NCBI
Point accepted mutation	PAM
Receiver operating characteristic	ROC
Ribonucleic acid	RNA
Ribosomal RNA	rRNA
Transport RNA	tRNA
True negative	TN
True positive	TP
True positive rate	TPR
Self-organizing map	SOM
Support vector machine	SVM
Thymine	T
Whole-genome shotgun sequencing	WGS

Chapter 1: Introduction

1.1 Introduction

Relatively little is known about the majority of microbes despite the fact that they are virtually everywhere. This is largely due to their resistance to standard cultivation techniques [1]. In the case of viruses there is also no known conserved genetic element shared among them all, making it impossible to study the total viral diversity through a single conserved genetic marker. However, learning more about viruses and other microbes is desirable and would be incredibly useful, as they are major players in global marine biogeochemical cycles and genetic exchange [2]. For example, in the ocean approximately 50% of the CO_2 fixed by photosynthesis each day ends up supporting the microbial community [3].

To overcome the limitations of standard cultured methods, microbial communities are being studied in an uncultured state. In such studies, genetic material is sampled from the environment, cloned, sequenced, and analyzed mathematically and algorithmically, as further described in Section 2.5. Rather than studying an individual genome, the collective microbial genomes of the community, called the *metagenome*, is studied.

The computational analysis of a metagenomics project can be divided into two main steps. The first step is the assembly of the fragments ideally into all the different genomes composing the microbial community. Current sequencing techniques normally target a single genome at a time. One example is the whole-genome shotgun sequencing technique (WGS). WGS produces an enormous amount of genome fragments. Afterward those fragments need to be assembled to get a clearer view of the genome. To do so, bioinformaticians have developed computer tools designed to help assemble these fragments into a continuous

genome. Although developed for single genomes, this sequencing method has been used to attempt to identify the entire metagenome of the microbial community. However, current assembly programs are optimized to assemble the genome of a single individual and not of an entire population, presenting a host of problems to be solved. It is clear that the assembly process needs to be adapted to the multi-genome problem.

In the assembly step it is unrealistic to think that we could completely reconstruct all the different genomes representing the biodiversity of the community. The number of sequences needed to cover all the different genomes is usually too large. Therefore, we will have, in most of the cases, only partial representations of the genomes. This fact will influence the subsequent step: the annotation.

1.2 The Problem: Annotation

The annotation process in a metagenome project seeks to identify assembled fragments (contigs) and the unassembled fragments (singletons) by comparing them to all known sequences found in a biological database. Current methods query the contigs and singletons against a database of known genes, such as GenBank[1], to find significant matches using a sequence comparison algorithm such as BLAST [4] further described in Section 3.1.

Metagenomics projects give only portions of the different genomes present in the community. Partial genomes have a high probability of having only portions of genes. The high number of incomplete genes composing a metagenomic project and the fact that we target unknown genomes of an unknown abundance and diversity lead to very poor results

[1]GenBank and other biological databases can be accessed here: http://www.ncbi.nlm.nih.gov/

in the crucial step of annotation. Some studies show that up to 75% of the sequences in the uncultured sample do not match anything in the database significantly, leaving those fragments without annotation [2].

1.3 Contribution of this thesis

In this context, the research described in this thesis seeks to annotate the metagenomic fragments before the assembly process. That is, we attempt to classify each fragment in a taxonomic structure. We thus aim to apply the machine learning methods of clustering and Hidden Markov Models (HMM), which have been used successfully in bioinformatics, to (1) reduce the number of unannotated fragments and (2) provide taxonomic annotation automatically. We show that the resulting system, which we call Anacle, leads to better results, in the sense that the annotation is more thorough and gives more information automatically than the current method.

Furthermore we hypothesize that this taxonomic annotation will help, in a recursive process, the assembly of the genomes by first grouping together fragments under the same taxa. This will improve the assembly process by restraining the number of fragments to be assembled, allowing an assembly adapted for the presence of polymorphism.

The work described in this thesis is part of a larger project being worked on by the Bioinformatics Laboratory (BiL) from the Information & Computer Science (ICS) department in collaboration with Dr. Grieg Steward from the Oceanography department at the University of Hawai‘i at Mānoa that aims to improve metagenome assembly and annotation methods.

Chapter 2: Biology Background

In this chapter we review the basic biological concepts of biological sequences and taxonomy.

2.1 Biological Sequences

There are three main classes of biological sequences implicated in the storage, conversion, transmission, and expression of genetic information: (1) Deoxyribonucleic acid (DNA), (2) Ribonucleic acid (RNA), and (3) protein. These molecules are polymers formed from smaller molecules in a sequential manner. DNA and RNA are composed of nucleic acids, while proteins are composed of amino acids. Table 1 shows the "alphabet" of each class of biological sequences. Since these sequences are linear, we can represent a sequence as a string over its respective finite alphabet.

The linear order of molecules of a sequence is called its primary structure. Higher levels of structure are determined by the biophysical interactions of the molecules the sequence is composed of and by the interactions of the sequence to the environment. That is, the primary structure ultimately determines what the physical 3D structure of the molecule will be, as illustrated in the example protein folding in Figure 1.

DNA		RNA		Protein	
Nucleic acid name	1-letter name	Nucleic acid name	1-letter name	Amino acid name	1-letter name
Adenine	A	Adenine	A	Alanine	A
				Cysteine	C
				Aspartic acid	D
				Glutamic acid	E
				Phenylalanine	F
Thymine	T	Uracil	U	Glycine	G
				Histidine	H
				Isoleucine	I
				Lysine	K
				Leucine	L
Guanine	G	Guanine	G	Methionine	M
				Asparagine	N
				Proline	P
				Glutamine	Q
				Arginine	R
Cytosine	C	Cytosine	C	Serine	S
				Threonine	T
				Valine	V
				Tryptophan	W
				Tyrosine	Y

Table 1. Alphabet of DNA, RNA, and protein sequences.
Full names of the nucleic or amino acid are listed along with their one letter shorthand name.

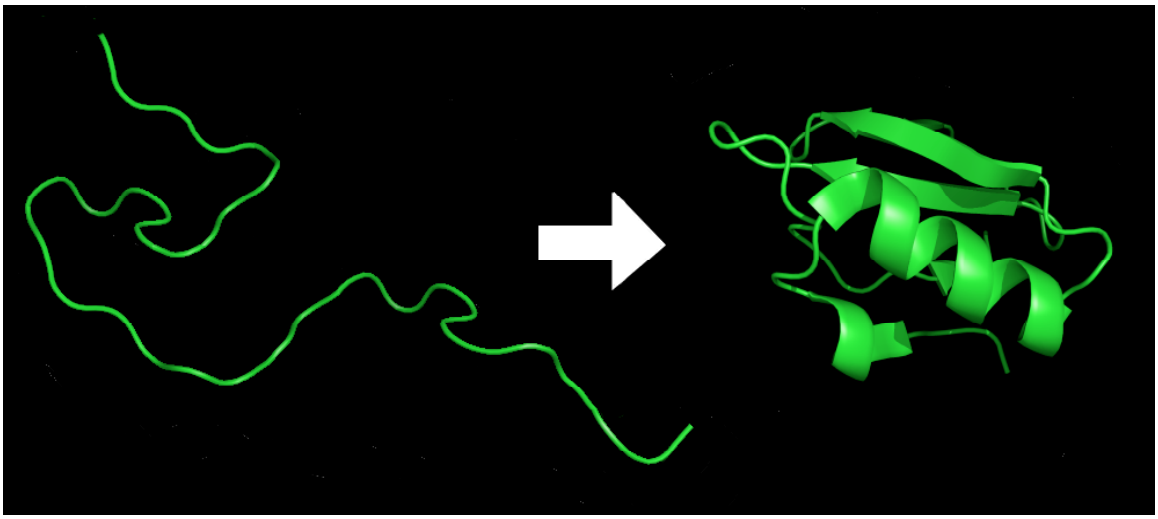

Figure 1. Illustration of protein folding.
This figure is from the public domain: http://en.wikipedia.org/wiki/Image:Protein_folding.png.

2.1.1 DNA and RNA

DNA was first isolated and discovered by Friedrich Miescher in 1869 [5] but it would not be known that it is the basis of heredity for a long time to come. In the early 1900's Phoebus Levene studied DNA extensively and discovered that DNA is a linked string of nucleotides [6]. There are four types of nucleotides in DNA: cytosine (C), thymine (T), adenine (A), and guanine (G). A strand of DNA can therefore be represented by a string over a four-letter alphabet. A small example would be a string like "ATCAATTG". It was then finally shown by Alfred Hershey and Martha Chase with their research on bacteriophages that DNA was indeed the genetic material that biologist have long sought to find [7]. Some viruses alternatively use RNA as their basis of heredity. RNA is very similar to DNA but with some differences, mostly notably of which is that thymine (T) is replaced with uracil (U).

The nature of how DNA could be replicated was not understood until James Watson and Francis Crick, using Rosalind Franklin's X-ray diffraction images, discovered double stranded DNA's (dsDNA) double helix structure [8] illustrated in Figure 2. They deduced that between the two stands, "A" always paired with "T"[2] and "G" always paired with "C". That is, the two strands are complementary. A way to replicate DNA now becomes clear: split the dsDNA into two separate strands and build the complement stand over both of them. The structure of DNA also tells us that it is directional, as denoted by what are called the 5′ and 3′ ends of the strand. That is, there is one correct way to read the sequence of nucleotides. For example if chemically we see 5′-G-A-T-T-A-C-A-3′ then we know that the

[2]Uracil (U) is the complement of adenine (A) for RNA strands.

strand is read from the 5′ end to the 3′ end as GATTACA rather than ACATTAG. Also the complement of this DNA strand would be 3′-C-T-A-A-T-G-T-5′, and so would be read as TGTAATC since it was shown that complementary strands of dsDNA are antiparallel.

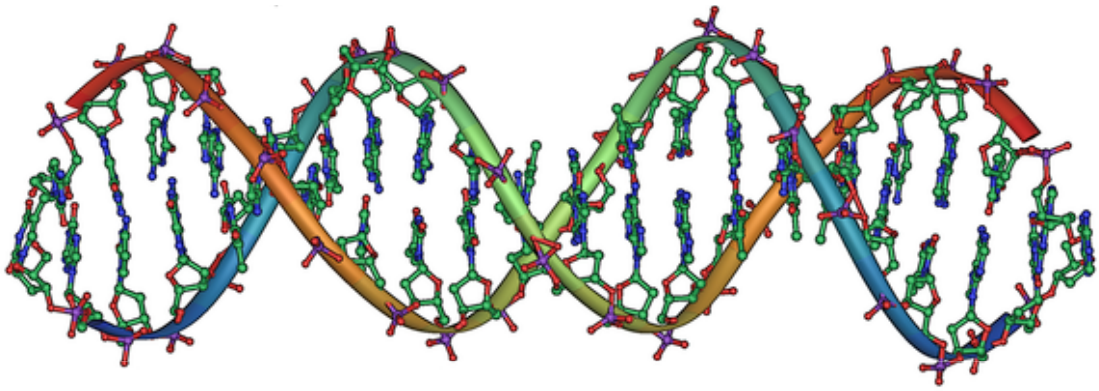

Figure 2. The structure of the DNA double helix.
Created by Michael Ströck. Released under the GFDL.

2.1.2 Protein Sequences

Proteins are perhaps the most important substances in an organism. Their functions have a wide range: the storage and transport of other substances, communication between different parts of an organism, to the defense of foreign substances [9]. Despite this diversity, all proteins are constructed from polymers of amino acids, called polypeptides. There are 20 amino acids and thus polypeptides can be represented as a string over a 20-letter alphabet (Table 1). The set of polypeptides of a protein interact chemically and fold into a 3D structure. It is this 3D structure that determines the protein's function.

As mentioned earlier, the final 3D structure is primarily determined by the protein's primary structure, which is simply its unique linear string of amino acids. One classic but extreme example of how a modification to the string can affect the final structure is

illustrated by hemoglobin. A single amino acid substitution at a particular position would turn the normally disk-shaped red blood cell into a sickle shape, a condition called sickle-cell disease [9]. While this is an extreme case but it illustrates the importance of the linear order of amino acids in determining the protein's 3D structure and thus function.

Unfortunately it is not easy to determine a protein's 3D structure and thus its function from just the primary structure. It has therefore become standard to infer protein function based on sequence similarity to a set of proteins of known function, as will be further described in Section 3.1. This works well, as the primary structure has a huge impact on the protein's structure.

2.2 From DNA to Protein

Biochemists have shown that proteins are the workhorse of organisms, providing the vast majority of an organism's functions and features. By contrast, geneticists have shown that DNA is the vehicle of inheritance—it is the physical substance that gets passed from parent to child. One may then find it surprising that the proteins themselves do not directly transfer from parent to child, but that only DNA does. Yet children are able to produce the same proteins as their parents and thus have the same traits. This caused some confusion in early biology and was finally solved with the discovery that genes in DNA can be translated into proteins [10]. Molecular biologists study the interaction of DNA, proteins, and other molecular substances and they provide the bridge between biochemistry and genetics.

There is an intermediary step in the complex biological mechanisms that translate DNA to protein: the RNA. There are different forms of RNA: messenger RNA (mRNA), ribosomal RNA (rRNA), and transfer RNA (tRNA). We are mainly concerned with mRNA

here. mRNA is created from the complement of the coding DNA in a process called transcription. Thus mRNA is a copy of the gene. This RNA is called messenger RNA, as this mRNA travels in the cell to the actual location in which the protein translation will occur. In summary, information in the cell is passed from DNA to RNA and finally from RNA to protein. This is often called the central dogma of molecular biology (Figure 3).

Figure 3. Central dogma of molecular biology.

Proteins are encoded into DNA, and thus RNA, by a non-overlapping triplet code called the genetic code. A triplet of nucleic acids is called a codon and its corresponding amino acid is known. Table 2 presents the 4^3 possible codons or 64 possible triplets. From those, 61 encode for different amino acids and 3 force the end of the process (stop codon). As a result of this, there are 3 possible frames of translation for each DNA sequence fragment, as illustrated in Figure 4. When we have a complete gene it is easy to deduce the resulting protein since we know exactly where the protein starts, i.e. the frame of translation, however for a sequence fragment, the frame of translation is unknown.

A DNA sequence usually encodes many proteins. In the case of dsDNA, both stands may have genes. So in the case of a sequence fragment, the origin of the strand is unknown so we need to consider 6 possible frames of translation (3 on one strand and 3 on the other) to deduce the 6 possible protein sequences.

		2nd base			
		U	C	A	G
1st base	U	UUU (F) UUC (F) UUA (L) UUG (L)	UCU (S) UCC (S) UCA (S) UCG (S)	UAU (Y) UAC (Y) UAA (Stop) UAG (Stop)	UGU (C) UGC (C) UGA (Stop) UGG (W)
	C	CUU (L) CUC (L) CUA (L) CUG (L)	CCU (P) CCC (P) CCA (P) CCG (P)	CAU (H) CAC (H) CAA (Q) CAG (Q)	CGU (R) CGC (R) CGA (R) CGG (R)
	A	AUU (I) AUC (I) AUA (I) AUG (M)	ACU (T) ACC (T) ACA (T) ACG (T)	AAU (N) AAC (N) AAA (K) AAG (K)	AGU (S) AGC (S) AGA (R) AGG (R)
	G	GUU (V) GUC (V) GUA (V) GUG (V)	GCU (A) GCC (A) GCA (A) GCG (A)	GAU (D) GAC (D) GAA (E) GAG (E)	GGU (G) GGC (G) GGA (G) GGG (G)

Table 2. Table of standard RNA codons.
This table shows the 64 codons and the amino acid each codes for. Recall that the nucleotides U and T are conceptually equivalent, so the above table can be used to translate DNA sequences also.
Note that the genetic code is not universal and may differ from species to species.

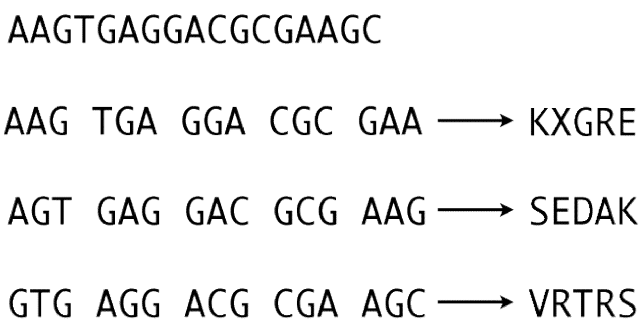

Figure 4. Three frame translation of DNA fragment.
This example uses the genetic code specified in Table 2 and uses the letter X to represent Stop.

Not all DNA are genes however, as there are sections of DNA with no known function. These noncoding DNA includes introns and intergenic DNA. In Eukaryotic DNA, introns are sections of DNA that are transcribed into RNA but later spliced out and missing from the final protein. The sections of DNA that produced the coding regions are called exons. The signals that mark the beginning of a gene and where the introns and exons lie are not

fully understood and is an area of open research that has lead to gene finding tools for use in genome projects. Thus the translation from DNA to proteins is a nontrivial matter. An example transcription of DNA to RNA and translation of RNA to protein is given in Figure 5.

DNA	TAC	CGC	GGC	TAT	TAC	TGC	CAG	GAA	GGA	ACT
mRNA	AUG	GCG	CCG	AUA	AUG	ACG	GUC	CUU	CCU	UGA
Protein	M	A	P	I	M	T	V	L	P	Stop

Figure 5. Example transcription and translation.

2.3 Linnaean Taxonomy

All living organisms are classified following many different taxonomic systems. The Linnaean taxonomy that is still popular today is be described here. With this system species are classified in a ranked hierarchy. The lowest rank contains the individual species such as humans, *Homo sapiens*. The next rank in the hierarchy groups similar species into genera (singular: genus). Then in the next rank similar genera are grouped into families, and so on, as shown in Figure 6. The highest levels of the Linnaean taxonomy have changed over time and more recent proposals split all life into three domains [11]: Archaea, Bacteria, and Eukaryota. Archaea and Bacteria are two broad divisions of prokaryotes, simple single cell organisms, while Eukaryota includes the more complex organisms such as those classified as animals and plants.

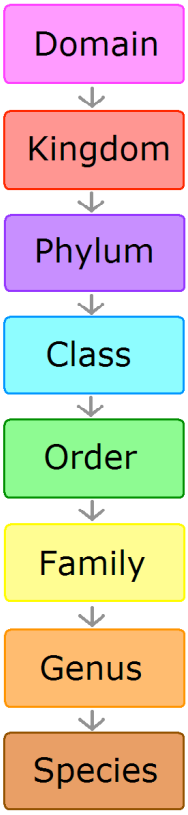

Figure 6. Linnean classification levels.

2.3.1 ICTV Taxonomy

Omitted from Linnaean taxonomy are viruses, as they do not fit the definition of "life." Viruses are not cellular and do not reproduce using their own machinery, instead relying upon their host organism for such functions. The International Committee on Taxonomy of Viruses (ICTV) has devised a similar ranked hierarchal classification similar to that of the Linnaean system. The ICTV system in fact uses the same naming scheme as the lower levels of the Linnean system. The highest level splits all viruses into different orders. Then each order is split into families, subfamilies, genera, and then finally virus species. Figure 7 illustrates the ICTV system with the species *Human herpesvirus 1* (HHV-1). Our annotation

system, Anacle, uses the ICTV virus taxonomy as it is popular and is used in sequence databases like GenBank to list a virus' classification.

Domain: Virus
↳ dsDNA virus, no RNA stage
↳ Family: Herpesviridae
↳ Subfamily: Alphaherpesvirinae
↳ Genus: Simplexvirus
↳ Species: *Human herpesvirus 1*

Figure 7. Classification of HHV-1.
Note that the taxon below the Domain rank is not marked as Order.
Herpesviridae is currently not classified under an Order.

2.4 Genomics

The genome of an organism is its complete hereditary information contained in DNA (or in RNA in some viral cases). As described in the previous sections, the genome contains all the information about what proteins the organism may construct and thus what functions the organism may express. Genomics is the study of an organism's entire genome rather than just a single gene. A major aspect of genomics concerns the sequencing of the genes composing the genome. That is, biologists take the physical DNA or RNA and with the help of molecular biotechnology produce the sequence representation over the four-letter alphabet (e.p. ATGCTTCA…). This text representation of DNA is thus very convenient, allowing efficient communication, storage, and manipulation of genetic data for scientists, particularly for computer scientists. Once the sequenced genome is available, scientists can then begin to analyze the genome, annotate the locations and identity of genes, find where the introns and exons are located, etc. Due to the sheer magnitude of genomic data, the sequencing and the analysis of a genome is only possible because of the advances in computer algorithms and technology.

Current limitations prevent sequencing machines from determining an individual's genome directly. The WGS technique was designed especially to help overcome those limitations. This method employs a random shearing of the organism's genome into millions of pieces of different lengths. The retrieval of the genome sequence from the many smaller sequences is called assembly. Conceptually, assembly is analogous to piecing together a jigsaw puzzle: the assembler must piece together the shorter sequences by searching for overlaps between them until the complete genome is constructed. Many algorithms and tools have been proposed to solve the assembly problem.

2.5 Metagenomics

Metagenomics, the application of modern genomics to the study of microbial communities directly in their natural environments, was born in 1985 with Pace's proposal of studying ribosomal RNA (rRNA) sequences of populations [12]. A metagenomics project begins with the retrieval of the genetic material of an environmental sample, such as from seawater or soil, and the construction of a clone library. With the great advances that have been made in sequencing technology, it is now feasible to sequence the entire clone library via WGS [13]. One of the first metagenomics projects that used this WGS approach studied two different marine communities [2] and there have been several others since, like the Sargasso Sea study [14].

Whereas the goal of a genomics project is to sequence one genome (an individual or of a single species), the goal of metagenomics is to sequence the genome of every species in the community. While Arachne and other assemblers are optimized for single genome assembly, such assemblers are being used for the multiple genomes assembly problem because there is

no alternative—a multi-genome assembler does not currently exist. Adapting a single-genome assembler to a multi-genome assembler brings about two issues that need to be overcome: 1) an increase in sequence polymorphism (DNA sequence differences between individuals of the same species) due to the use of fragments originating from different individuals in the population and 2) highly conserved sequences between species leading to false overlaps in the assembly process. Because of these issues, the results of running a single-genome assembler must be manually processed and corrected. The larger project that this thesis is under aims to make improvements to this multi-genome assembly process including the removal of this manual step.

The result of the assembler in a metagenomics project is a set of scaffolds or supercontigs, which are partially assembled fragments. To get a sense of what sort of organisms are contained within the sampled community, the scaffolds need to be categorized as specifically as possible. This is the metagenome annotation problem this thesis studies and that will be further described in subsequent chapters.

Chapter 3: The Traditional Annotation Process

This chapter reviews the traditional annotation process and a related method for metagenomics sequences. One important notion related to the annotation process is the similarity of sequences. This chapter also introduces this concept.

3.1 Sequence Similarity

Pairwise sequence similarity is a measure of how related two protein or DNA sequences are. This measure is usually based on a pairwise alignment of the two sequences. An example similarity measure would be the percent identity, the percentage of identical residues (amino acids or nucleotides) that line up with each other in the alignment. Such measures can be used to quantify evolutionary changes or identify residues crucial to the protein's structure and function.

Percent identity however does not suffice and more sophisticated methods have been developed to not only score matching residues but also to score residue substitutions, insertions, and deletions. The score for a particular substitution is calculated empirically through observations of substitution frequencies. Examples of scoring matrices for proteins are the PAM (point accepted mutation) [15] and BLOSUM (blocks substitution matrices) [16] matrices.

The calculated similarity score of two sequences is then dependent on the alignment of the two sequences. Different alignments may lead to different similarity scores. It is up to algorithms to find the optimal alignment, the alignment that leads to the maximum similarity score. Dynamic programming algorithms have been formulated to solve this problem, but

heuristic algorithms that find approximate solutions are used in practice for their sheer speed. There are two types of optimal alignment and thus two types of sequence alignment algorithms. The first is known as global alignment, where the optimal alignment and score is found by considering the entirety of both sequences. An example is the Needleman-Wunsch algorithm [17] that uses dynamic programming. The other type of alignment is known as local alignment that calculates the optimal alignment and score of subsequences of the two query sequences. It is up to the algorithm to find the subsequences that lead to the highest similarity scores. An example dynamic programming algorithm is the Smith-Waterman algorithm [18]. An example global and local sequence alignment is illustrated in Figure 8.

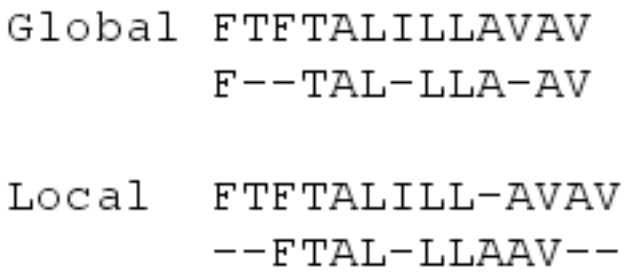

Figure 8. Illustration of global and local alignment.

BLAST (Basic Local Alignment Search Tool) is the most widely used technique for calculating sequence similarity. BLAST uses a heuristic algorithm to calculate the optimal local alignment [4]. The output of BLAST against a database of sequences returns the top hits of the query sequence, reporting for each the score, expectation value, and the local alignments themselves. The expectation value (*E*) provides a statistical measure of the significance of the alignment and score (*S*). *E* reports the expected number of hits having a score of *S* or more by chance. Low *E* values imply biological significance, while high values imply false positives [19].

3.2 Traditional Method: Annotation by BLAST

The result of the assembler in a metagenomics project is a set of partially assembled fragments. Now to get a sense of what sort of organisms are contained within the sampled community, these fragments need to be categorized as specifically as possible.

Metagenome annotation relies on the fact that prokaryotes have a high gene density and therefore current read lengths will likely contain a significant portion of at least one gene [20]. Thus if a gene on a fragment is a known gene or is closely related to one, the fragment should match closely in sequence to the known gene's sequence in a database. If the matched gene is known to be unique to a domain, family or species of microbes, it can be inferred that this is where the fragment originated. However a new sequencing technique, Pyrosequencing, generates fragments of only 100 nucleotides compare to the traditional 700-800 nucleotides. Pyrosequencing has the advantage of being cheaper and faster than traditional sequencing methods, allowing for a more through sequencing coverage of the metagenome. However, these short pyrosequences have a very low chance of containing an entire gene, making the annotation process even harder. Thus there are tradeoffs between the different sequencing methods.

The current approach is then to compare each fragment against GenBank, an open access and annotated sequence database, using BLAST. An expert can then manually infer the origins of a fragment using the top hits of the BLAST query.

This approach showed that much of the diversity in an uncultured community is uncharacterized, as about 75% of the sequences have no significant matches to sequences in GenBank [3]. However some of the unclassified sequences may actually be similar to known genes in the database, and were simply missed because of the present partial genes or the

limitations of a tool like BLAST. Also considering that a metagenomics project can currently produce about a half of a million or more fragments and that future projects will produce much more as sequencing cost decreases, it may soon become infeasible to annotate without automation. Including taxonomic information into the annotator will lead to better and more automated results.

3.3 Related Method: PhyloPythia

PhyloPythia is a recently published system that classifies DNA fragments taxonomically [21]. The system is able to automatically and taxonomically annotate fragments. Taxonomic information is integrated into PhyloPythia through the use of multiclass support vector machines (SVMs) at each rank. The number of classes at each rank varies, with for example the top rank of Domain consisting of three classes: Eukaryota, Bacteria, and Arachea. Note that PhyloPythia does not currently support viral taxonomy and thus will not be able to annotate virus fragments. Since SVMs are binary classifiers, each rank consists of $N(N-1)/2$ distinct pairs of SVMs (one for each possible pair of taxa), where N is the number of taxa in the rank. A voting mechanism among the SVMs decides which taxon to assign the fragment to. A final one-versus-all SVM is then run to detect and discard false positives. This is very computationally expensive due to the sheer number of SVMs that need to be trained and queried.

Phylopythia classifies at the DNA level omitting the more informative protein stage. Also this method works better with longer fragments or even contigs and was not tested on pyrosequences. However this method shows that the addition of the taxonomic information in the annotation process clearly increases the number of annotated sequences.

Chapter 4: Clustering Methods

In this chapter we review the clustering methods of Self-organizing Maps and Markov clustering, particularly as related to the protein clustering problem. These techniques are used by the new skeleton annotation method, as will be described in Chapter 5.

4.1 Protein Clustering

The goal of cluster analysis, in general, is to group a set of objects into subsets, or clusters, such that the objects within each cluster are more similar to each other than to objects belonging to different clusters [22]. The goal of protein clustering methods is then to group proteins that share, for example, similar functions or similar sequence motifs together while separating them for those proteins that are dissimilar. The notion of similarity must be explicitly defined in order for a clustering method to be formulated, and the measure often used in protein clustering is the sequence similarity score, as described in Section 3.1, from a tool such as BLAST. Alternatively, a protein can be represented by a set of numerical measurements, such as those described in Section 4.2, and a metric such as Euclidean distance can be used as the measure of similarity.

It is common for a clustering method to require that the user specify the number of clusters. It is also often the case, as in this thesis research, that the number of clusters is not known. The determination of the number of clusters given a dataset is recognized as one of the most difficult problems of cluster analysis [23]. It is common practice then to use heuristics, for instance via an additional criterion like the GAP-statistic [24] or via cross-validation methods, to determine the maximal number of clusters present in the data.

For these reasons, in choosing the clustering methods for this thesis research it was important that the methods do not require the number of clusters to be explicitly specified and that the methods have been previously shown to produce biologically meaningful clusters. The two methods investigated and described in this chapter are Self-organizing Maps (SOM) and Markov clustering (MCL). As described in Section 4.2 and 4.3 SOMs require that the proteins be represented by a vector of numerical measurements, while MCL, described in Section 4.4, uses sequence similarity as reported by BLAST. Thus these two methods are quite different. MCL was chosen for the majority of this thesis research's experiments due to its superior running time speed, better multiple alignments of the members of a cluster, and better annotation results, as discussed in Section 5.3.

4.2 Protein Representation

In order for a dataset of proteins to be clustered using SOMs or some of the other clustering methods, the proteins must be represented or encoded by a vector in some chosen feature space. For example protein representations based on dipeptide frequencies, further described below, can be used. These representations based on frequencies are an example of protein encodings that do not preserve the original amino acid sequence. Such sequence representations are called indirect encodings. Alternatively representations that preserve the original amino acid sequence could be used and are called direct encodings. However proteins exists in a variety of sequence lengths, and thus direct encoding over the entire protein length will lead to feature vectors whose dimension vary from protein to protein in the dataset. This is a problem for methods like SOMs that require a fixed input dimension. This problem could be overcome if the direct encoding of some fixed length subsequence,

such as the last 50 amino acids, that we knew was sufficient enough to solve the original problem was taken instead. This is not the case for this thesis research however, and only indirect encoding will be further described.

It has been shown that indirect encoding based on dipeptide frequencies leads to meaningful clustering of protein sequences into families [25]. An example protein sequence represented with such encodings is illustrated in Figure 9. The largest of such encodings is the straightforward dipeptide count resulting in a 400-dimensional (20x20) input vector. Furthermore, the input vector should be normalized to transform the representation to a percent composition. Since proteins come in many lengths, percentages are more useful than raw counts.

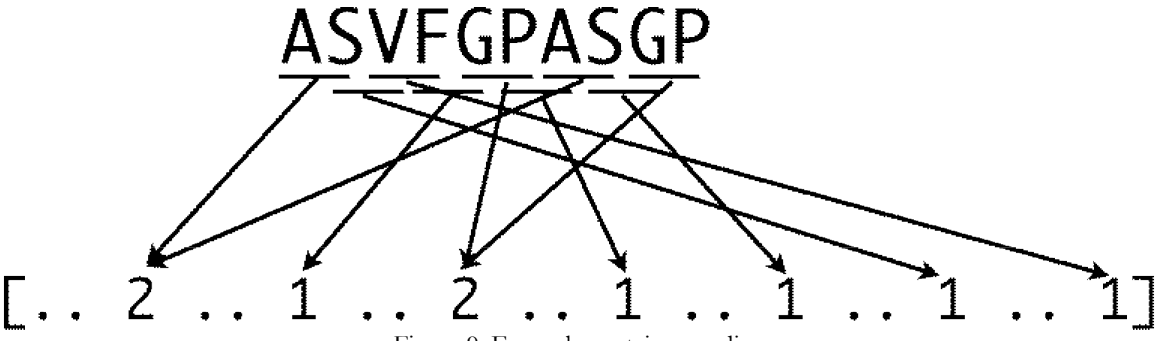

Figure 9. Example protein encoding.
The frequency counts of all ordered pairs of amino acids are taken. Thus the encoding is 400 dimensional.
The pairs AS and GP occur twice each and so their components are set to 2.
FG, PA SV, SG, and VF occur once. All other possible ordered pairs are set to 0.

Smaller encodings based on dipeptide counts can be created by grouping the 20 amino acids into related groups based on common properties such as hydrophobicity. An example encoding would split the amino acids into 11 groups and count the frequencies of ordered pairs of these groups, resulting in a 121-dimensional (11x11) input. The eleven groups are: {V, L, I}, {T, S}, {N, Q}, {E, D}, {K, R, H}, {Y, F, W}, {M}, {P}, {C}, {A}, and {G}. It

was shown that this representation leads to clustering results very similar to the 20x20 representation with the added benefit of having about a 3.3 reduction factor in computing time [25]. Another encoding uses six groups: hydrophobic {V, L, I, M}, hydrophobic aromatic {Y, F, W}, neutral/weakly hydrophobic {P, A, G, S, T}, hydrophilic acid {N, Q, E, D}, hydrophilic base {K, R, H}, and crosslink forming {C}, leading to a 36-dimensional input. Experiments indicate that the 36- and a smaller 9-dimensional encodings give fairly similar clustering results, but do not perform as closely to the larger encodings [25]. Similarly to the 400-dimensional encoding, these smaller encodings should also be normalized.

4.3 Self-Organizing Maps

SOMs are a type of artificial neural network (ANN) originally developed by Kohonen [26]. Unlike MCL, a SOM requires that the protein dataset be represented by fixed-length numerical vectors, such as the encodings described in Section 4.2. The measure of similarity used by SOMs is the Euclidean distance between feature vectors. The result of the SOM algorithm is a topology-preserving mapping of the dataset constructed using competitive learning. That is, the SOM maps, in a nonlinear fashion, the original feature vectors to vectors in another feature space of smaller dimension than the original feature space. The mapping is topology-preserving in that relative proximity of the data is the same in the mapped feature space as the original feature space. The mapped feature space is usually 2D so that the dataset is more easily visualized. If a small number of clusters are expected, the clustering may be easy to see visually with a 2D SOM. Otherwise a clustering method may be applied to the mapped feature vectors to find the clusters.

A SOM consists of two layers of units: the input layer and the competitive layer. The competitive layer is usually organized in a 2D grid of units, as illustrated in Figure 10. Each unit in the competitive layer is linked to every input unit. A competitive learning algorithm to construct the mapping is summarized below [27].

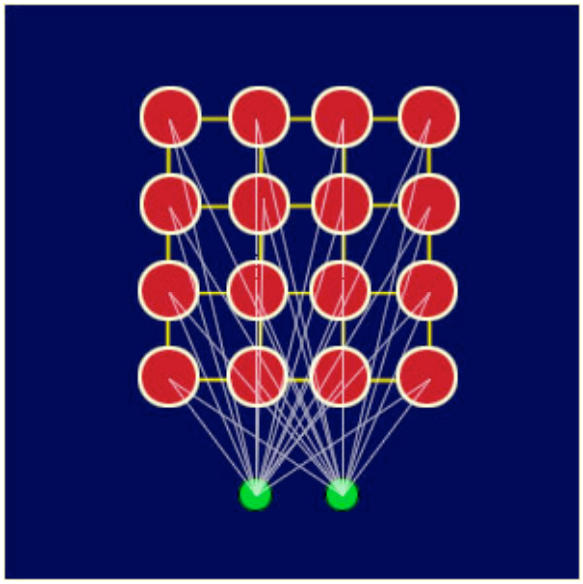

Figure 10. A simple SOM.
The competitive layer is 4x4 units and the input layer is two dimensional, represented by the two smaller green units below.
Graphic taken from: http://www.ai-junkie.com/ann/som/som1.html

1. *Initialization*: Let $\mathbf{w}_j$ denote the weight vector of the *j*-th unit in the competitive layer. Randomly choose initial weights $\mathbf{w}_j(0)$ for all competitive units.

2. *Similarity matching*: Draw a sample $\mathbf{x}$ from the dataset. Find the best-matching unit (BMU) $bmu(\mathbf{x})$ at time *n* using the minimum Euclidean distance criterion:

   $$bmu(\mathbf{x}) := \arg\min_j \left\| \mathbf{x}(n) - \mathbf{w}_j(n) \right\|$$

3. *Weight update*: Update all weight vectors using the following formula:

   $$\mathbf{w}(n+1) := \mathbf{w}(n) + \eta(n) h_{bmu(\mathbf{x})}(n)(\mathbf{x}(n) - \mathbf{w}(n))$$

where $\eta(n)$ is the learning-rate parameter and $h_{bmu(\mathbf{x})}(n)$ is the neighborhood function centered around the BMU. The learning-rate and neighborhood are usually decreased after each iteration or epoch.

4. Repeat steps 2 and 3 until convergence of weight vectors.

The goal of the neighborhood function is to make the mapping topology preserving by affecting the update of the weight vectors of units closer to the BMU more than the weight vectors of units further away. The finding of the BMU and learning of the units in the "neighborhood" of the BMU is the competitive learning used by SOM algorithms. An example neighbohood is illustrated in Figure 11.

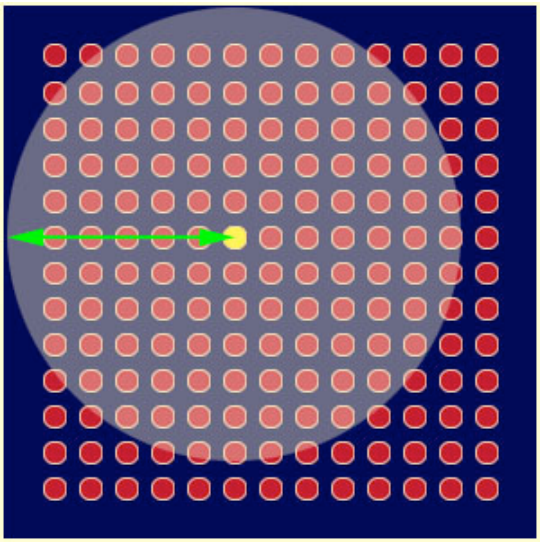

Figure 11. Example neighborhood of a BMU.
Units closer to the center unit, the BMU, are updated more strongly than the units further away.
This is called the Gaussian neighborhood.
Graphic taken from: http://www.ai-junkie.com/ann/som/som3.html

Upon completion of the construction of a SOM, we now have a mapping from the samples in the dataset to the BMUs of the samples. As mentioned above, the BMUs of the

samples can be used as a visualization of the dataset, where the clustering of the dataset may be easily seen by eye or may be computed using another clustering method.

4.4 Markov Clustering

MCL is a graph clustering algorithm that has been shown to be applicable to protein clustering [28]. A node in the graph represents a protein in the dataset. The weight of the edge between two nodes represents the similarity between the two proteins. The weight assigned to an edge is the average $-\log_{10}(E)$ leading to a symmetric matrix representation of the graph. Thus MCL uses the BLAST expectation value E as the measure of similarity between two protein sequences.

The graph's matrix is then turned into a Markov chain by normalizing the weights column-wise, resulting in a stochastic matrix **M**. Row entry *i* in column *j*, $\mathbf{M}_{ij}$, is the probability of transitioning from node *j* to node *i*. The weights can now be viewed as transition probabilities where the probability of transitioning to a highly similar node is larger than that of a transition to a less similar node. The aim of the MCL algorithm is to augment "flow", i.e. the number of random walks, within a cluster and eliminate the "flow" between clusters. This is accomplished using the following algorithm:

1. *Expansion*: Square the stochastic matrix **M**. The resulting matrix is still a stochastic matrix.

   $\mathbf{M} := \mathbf{M}^2$

2. *Inflation*: Raise each weight of **M** to the *I*-th power and then normalize the resulting weights column-wise. The normalization ensures the new matrix **M** is still stochastic. The inflation value *I* is the only parameter of this algorithm. Essentially the *I* value

indirectly determines the number of clusters. Formally each matrix is updated with the following formula:

$$\mathbf{M}_{pq} := \left(\mathbf{M}_{pq}\right)^{I} \Big/ \sum_{i} \left(\mathbf{M}_{iq}\right)^{I}$$

3. Repeat steps 1 and 2 until convergence of matrix **M**.

The expansion step corresponds to computing random walks of higher length, while the inflation step has the effect of boosting intra-cluster walks and demoting inter-cluster walks [28]. Upon completion of the MCL algorithm, the connected components of the final graph correspond to the individual clusters. This process is illustrated in Figure 12.

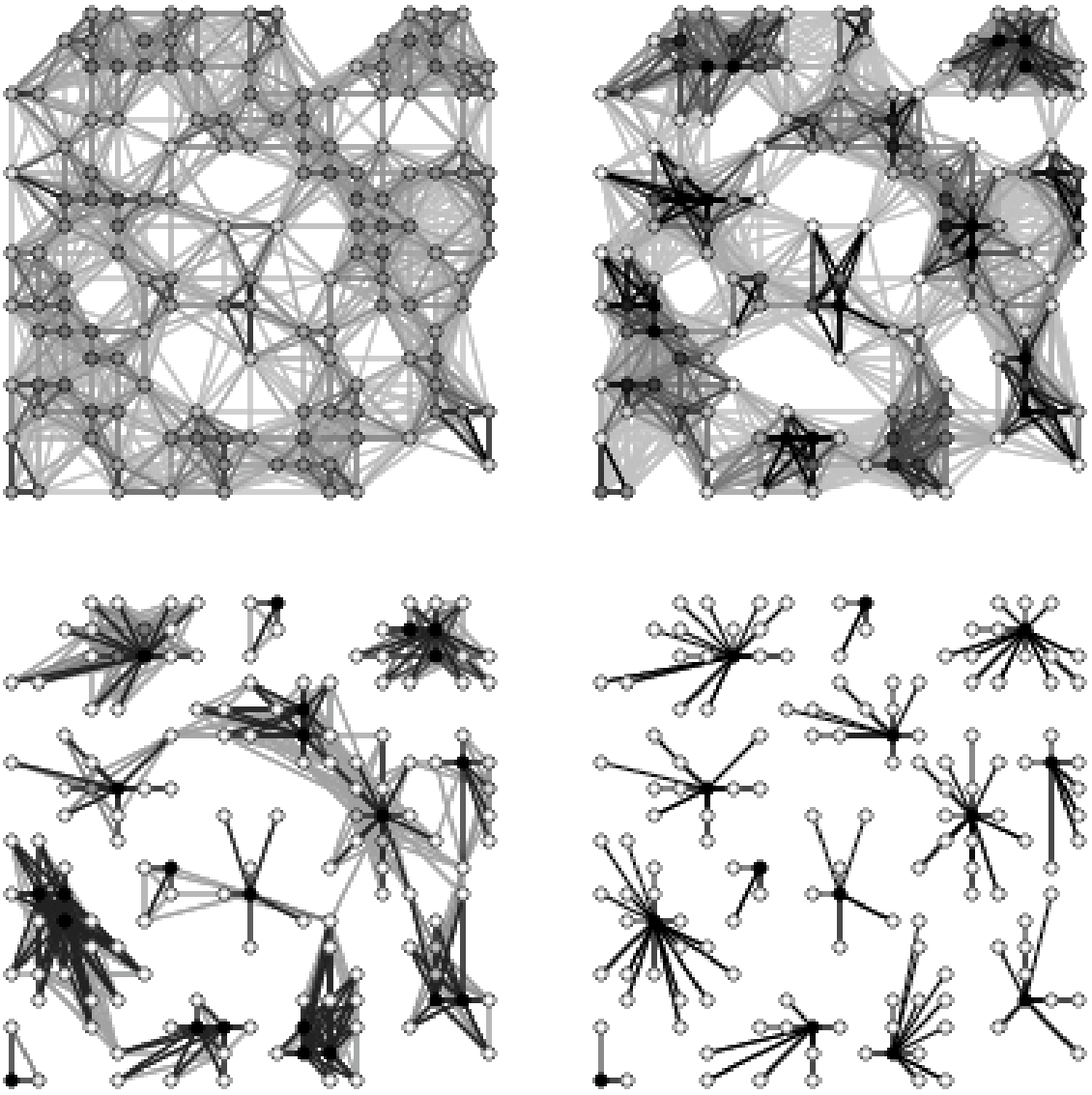

Figure 12. Visual MCL example.
The top left subfigure illustrates the initial graph. The darker edges represent close similarity between nodes, while lighter edges represent less similarity between nodes. Iterations of the MCL algorithm strengthen and weaken edges till convergence. The final graph on the bottom right shows the final clustering.
Figure taken from [29].

Chapter 5: The Skeleton Method

In this chapter we describe our work and the main contribution of this thesis, the skeleton method. The skeleton method is described and we present the experimental evaluation of the method's performance.

5.1 A New Method: Annotation by Skeleton

Knowledge of what genes exist together in certain taxa would help the classifier create better annotations. This thesis research seeks to integrate such information into the annotator by constructing profile "skeletons" for different taxa. These skeletons consist of profiles of proteins, called profile Hidden Markov Models (profile HMMs), that are known to be found in the skeleton's taxon. The use of the protein sequences instead of the DNA sequences allows us to take advantage of the more informative stage represented by the protein and also gives less weight to sequencing error that are very common in DNA sequences from metagenomics or any sequencing project.

5.1.1 Profile Hidden Markov Models

A critical part to the new annotator, which we call Anacle, is clearly the profile HMMs that represent each protein. Profile HMMs are already commonly used in bioinformatics to represent the profile of a protein. In brief, HMMs are probabilistic models. There is an underlying model of states that is unobservable (hence the term "hidden" in HMM) and above that, each state has a probability of emitting observable events. The HMM can be thought of as a stochastic machine that generates a sequence of symbols over time. In the

case of profile HMMs, the symbols are amino acids and the generated sequence is the protein. One of the first uses of profile HMMs in computational biology was presented by Krogh and collaborators in 1994 [30]. Multiple sequence alignment is widely used to find functional and structural information important in the definition of a family of protein. The use of HMMs helped this task by allowing the use of position-specific score models and has been implemented in the software package called HMMER [31]. The profile HMM architecture used by HMMER is shown in Figure 13. The squares indicate match or consensus states (M#) that model highly conversed residues. Diamonds indicate insertion states (I#) and random sequence emitting states (N, C) that model additional residues before, after, and between consensus residues. Finally circles indicate delete states (D#) and begin/end states (S, T). The delete states models the deletion of consensus residues. Each state transition (arrows) has a probability associated with it. HMMs, however, can be used for more than just modeling a protein profile. HMMs have found widespread and successful use in bioinformatics, including such areas as gene finding, genetic linkage mapping, and protein secondary structure prediction [31]. HMMs have become an essential tool in bioinformatics.

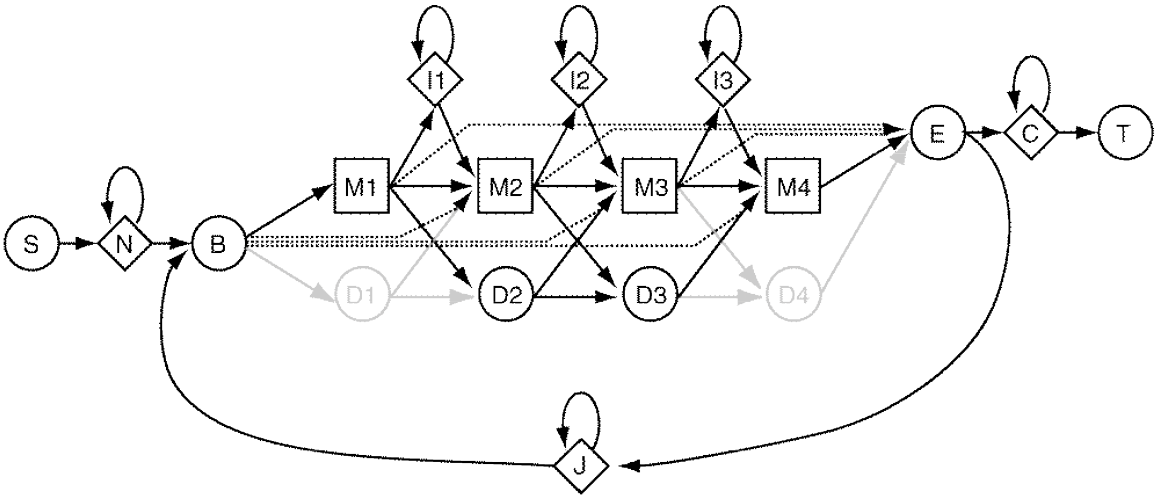

Figure 13. HMMER's profile HMM architecture.
From HMMER user manual: http://hmmer.janelia.org

However, the probabilities on which the models rely are not generally known and therefore must be estimated using multiple alignments of known representations of the protein in the case of profile HMMs or by using supervised machine learning. Once a profile HMM has been created, it can be used to calculate the estimated probability that a given sequence was generated by the HMM. That is, the likelihood that the given protein sequence is the same protein as the profile can be calculated, and this likelihood also serves as a degree of confidence.

Unlike pairwise comparison methods like BLAST, any number of sequences can be used to construct profiles. This allows more information, including the positions more conserved than others and different tolerances to insertion and deletion from region to region, to be used during comparison. This position specific information has lead to methods to better detect more distantly related proteins and improves the results of searching databases for homologous sequences [32]. Anacle, through the use of profile HMMs along with the

higher-level taxonomic information provided by the skeletons, should decrease the number of unannotated sequences and provide a more precise and automated annotation process.

5.1.2 Skeleton Construction

The first step in constructing the skeleton of a taxon is to find the genetic commonalities of all the known member species of the taxon. For example to construct the skeleton of the virus family *Herpesviridae*, we would first need to analyze the genomes of all known herpesviruses. As specified before, for the construction of the skeleton we use information at the the protein level. So we are interested in the proteins shared among some or all of the herpesviruses. That is, we want to divide the protein products of all herpeviruses into groups with similar proteins in the same group and dissimilar proteins in different groups. This grouping or clustering can be accomplished with the methods of cluster analysis described in Chapter 4. This analysis can be done with any other taxa, including but not limited to other virus families, genera of any type of organism, orders, etc. All known sequenced genomes and their genes and protein products can be found in databases such as GenBank.

Completion of the protein clustering step leads to a number of groups or clusters of proteins that represent the desired taxon. Each cluster can then be summarized and modeled with a profile HMM. The profile HMM of a cluster can be constructed through unsupervised machine learning, such as the simulated annealing Viterbi algorithm implemented in the program `hmmt` of the software package HMMER 1.8.5. Alternatively the profile HMM may be constructed from the multiple alignments of all member proteins of the cluster using, for example, the tool ClustalW. We chose to use the unsupervised learning provided by HMMER rather than the multiple alignment alternative due to the

difficulty in obtaining good alignments. In any case, the resulting profile HMMs, one for each cluster, represent the commonalties of all the members of the taxon. That is, these profile HMMs provide a summary and model of the genetic elements found in the taxon. It is this set of profile HMMs that we call the skeleton of the taxon.

5.1.3 Querying the Skeletons

The set of fragments of a metagenomics project can be queried against the skeletons of all taxa. The resulting output would be a score (likelihoods) for each fragment to all profile HMMs of every taxon skeleton. Naturally the profile HMM for which the fragment has the highest score, that is the profile HMM with which the fragment has the highest probability of membership, is the profile the fragment putatively belongs to. The taxonomic origins of the fragment can then be inferred from the taxon in which the highest scoring profile HMM belongs to. Thus the fragment is annotated taxonomically. If however the fragment scores low on all profile HMMs and thus is not likely to belong to any of the profile HMMs, we have no choice but to annotate the fragment as coming from unknown origins.

Alternatively, we may want to give more bias toward annotating a fragment with a lower ranking taxon. For example, even if a fragment's top hit is *Herpesviridae*, we may want to say that the fragment is from the subtype simplexvirus, which may have scored lower. This can be desirable since lower ranking taxa give more information about the fragment's origin than higher ranking taxa. This can be done by querying a fragment bottom-up, from lower ranking taxa to higher ranking taxa. Fragments that have high scoring hits on a lower taxon can be annotated as such, and the remaining fragments with low scores or not hits at all can

then be queried on the taxa in the next rank up. This can save a lot of computational time, as each fragments does not need to be tested against on all HMMs.

Since we use the protein sequences in the construction of the skeleton we need to translate all DNA fragments into its 6 possible frames of translation, as was described in Section 2.2. Our implementation of this analysis queries the 6-frame translated DNA fragment database to each profile HMM using the `hmmsw` program of HMMER 1.8.5. The resulting HMMER reports are then parsed to generate a report listing the top profile HMM hits for each fragment.

While annotation by skeleton is more computationally complex, especially in the initial skeleton construction, than annotation by BLAST, this new method does provide fuller, taxonomically based annotation that BLAST is incapable of producing.

5.2 Dataset

The Integrated Microbial Genomes (IMG) database[3] offers the complete DNA genome, along with all known protein sequences, of all sequenced virus genomes. Using the lineage listed in the database, the genomes can be grouped taxonomically and can then be used to build skeletons for selected or all taxa. These protein sequences were used to build our skeletons.

To evaluate the taxonomic annotation process, we build a simulated metagenome. To simulate a metagenomics project, a selection of virus genomes are taken and a number of random fragments are taken from their DNA. The skeleton annotator can then take these

---

[3]IMG website: http://img.jgi.doe.gov/

fragments and classify each one taxonomically. Unlike a true metagenomic dataset, we know the true origin of each fragment and thus we are able to evaluate the performance of the annotator. The length of the DNA fragments is set to either 700 basepairs (bp) or 100 bp. The length of 700 bp is the typical length of a fragment when using standard DNA sequencing, while 100 bp is the typical length for Pyrosequencing, the fastest sequencing method to date. Note that the majority of the preliminary tests were done on the 700 bp datasets. We afterward confirmed the possibility of annotating 100 bp fragments by redoing some of the tests using the 100 bp datasets. The final tests done on all the possible virus taxa were done for both fragment lengths.

Since the skeleton is trained using protein sequences, the DNA fragment dataset needs to be translated into protein fragments using the genetic code. As shown in Section 2.2, the translation of a DNA fragment is however ambiguous, as we do not know where the non-overlapping code begins. The first nucleotide of the fragment may not necessarily be the first nucleotide in the codon, it could be the second or last. That is, we do not know the correct reading frame. With this consideration then, we end up with three protein translations of the DNA fragment. Since the protein may be encoded on the complementary strand of dsDNA, we must also translate and consider the three additional protein translations from the DNA fragment complement. Among these six translations is the true translation from the correct reading frame. In our simulated metagenome the correct translation of a DNA fragment can be determined by using BLAST.

It should also be noted that Anacle currently assumes the standard genetic code (Table 2) in its protein translations, and that the inclusion of alternate genetic codes is a future extension would most definitely improve annotation results.

A script using BioPerl[4] was written to generate the artificial metagenome by cutting random DNA fragments from a set of genomes and translating the six frames into protein sequences.

5.3 Clustering Methods: SOM and MCL Skeletons

We evaluated two different methods of clustering for the construction of the skeletons: the SOM and MCL methods. The SOM clustering of proteins was done using a Matlab library[5] that also clustered the resulting BMUs via linkage (a variety of linkage options is given and used for the evaluation). The MCL clustering was done using the successor of TribeMCL in the C-implementation of MCL [28]. A script was written to generate protein sequence files for each resulting cluster, which is then used to generate HMMs using HMMER [31].

It is unclear what clustering method and parameters would provide better taxa skeletons without doing some experimentation. This set of experiments aimed to determine what method and parameters looked the most promising for use in the next sets of experiments.

5.3.1 Experimental Design

All known proteins of a subset of the virus family *Herpesviridae* were used for clustering using SOM and MCL under a variety of parameters. The resulting clustering was used to generate the HMM skeleton for *Herpesviridae*. The same subset of herpesviruses along with

[4] BioPerl available at http://www.bioperl.org
[5] SOM Toolbox developed by the Laboratory of Computer and Information Science Adapative Informatics Research Centre: http://www.cis.hut.fi/projects/somtoolbox/

other viruses outside the family was used to generate the test metagenome. The test metagenome consists of fragments each of length 700 bp. For each genome 50 fragments were generated. Appendix A.1 lists the genomes used for training and for the fragment generation.

Two SOMs are trained, one using the 11x11 protein encoding and the second using the 20x20 encoding, with a 40x40 unit competitive layer, as described in Section 4.2. For each trained SOM, the BMUs were clustered using all available linkage options given in the SOM toolbox: single, complete, average, centroid, ward, neighf, and closest. This results in seven different clustering results per SOM. The annotation performance of each resulting clustering was tested. For MCL, we clustered using a range of inflation values. The inflation value essentially determines the number of clusters and is MCL's one and only parameter.

5.3.2 Results

The average top hit score and hit percentage of a genome's fragments are the statistics used for the comparison. The average top hit score and hit percentage of a *Herpesviridae* genome should be high and close to 100%, respectively. Ideally for a non-*Herpesviridae* genome, the average top hit score and hit percentage should both be low. The score HMMER reports is the log-odds score,

$$S = \log_2 \frac{P(seq \mid HMM)}{P(seq \mid null)},$$

where $P(seq \mid HMM)$ is the probability of the target sequence according to the profile HMM and $P(seq \mid null)$ is the probability of the sequence given the null hypothesis that the sequence is random [31]. Since the log is base two, the score is in units of bits. We are interested in hits with high positive scores, which imply that the sequence is highly similar to those hits.

The skeletons constructed using a SOM with a variety of parameters resulted in very similar performance, as the type of linkage made little difference. As such, we will only show the results from the skeletons constructed using the 11x11 and 20x20 encodings and clustered using complete linkage. When clustering with MCL, it was observed that an inflation value below 1.2 resulted in a few clusters that were too large. The larger clusters would contain proteins that were unrelated in terms of function and sequence, as determined by known annotation and multiple sequence alignment. For example, the results of clustering the proteins of the herpesviruses with an inflation value of 1.1 contains one unusually large cluster with 223 members (the next largest cluster only has 73 members) that contains proteins from a variety of different functions, from translation regulation to capsid assembly and transport. On the other hand, larger inflation values lead to too many clusters, leaving many singleton clusters where it was observed that proteins that were similar were not grouped together. For herpesviruses, an inflation value of 1.3 leads to 54% of the clusters being singletons. It is observed that an inflation value of 1.2 leads to more balanced results where for herpesviruses the largest cluster contains 20 members and the number of singleton clusters is reduced to 50%. This is the value used in the MCL results below.

Figures 14 and 15 compare the average top hit scores and hit percentages for the skeletons generated by MCL and SOM clustering. In Figure 14 we can see that there is no clear advantage between using the 11x11 and 20x20 protein encodings in the case of SOMs. All three clusterings shown offer similar, good performance with high scores for *Herpesviridae* fragments and low scores for non-*Herpesviridae* fragments. There is an apparent advantage of using MCL over SOM, as the average top hit scores in general are higher than those from the SOMs for *Herpesviridae* genomes and lower for non-herpesviruses.

Figure 15 shows that around 90-100% of a herpesvirus' fragments earned hits for both clustering methods, indicating that the skeleton is detecting herpesvirus fragments well. The percentage of hits among the non-*Herpesviridae* genomes is a mixed bag, with some genomes being low and some being as high as 100%. MCL shows an advantage here again with lower hit percentages than the SOMs. Since the difference between the score of a herpesvirus fragment and a non-herpesvirus fragment is large on average, we can eliminate many false positive hits with an appropriate threshold score. Thus regardless of the encodings and clustering methods tested, good results are observed. This may indicate that the skeleton method is not very sensitive to different clusterings. Since the MCL results show some advantages and the running time of MCL is orders of magnitude faster than training a SOM[6], we choose to run the next experiments using MCL exclusively.

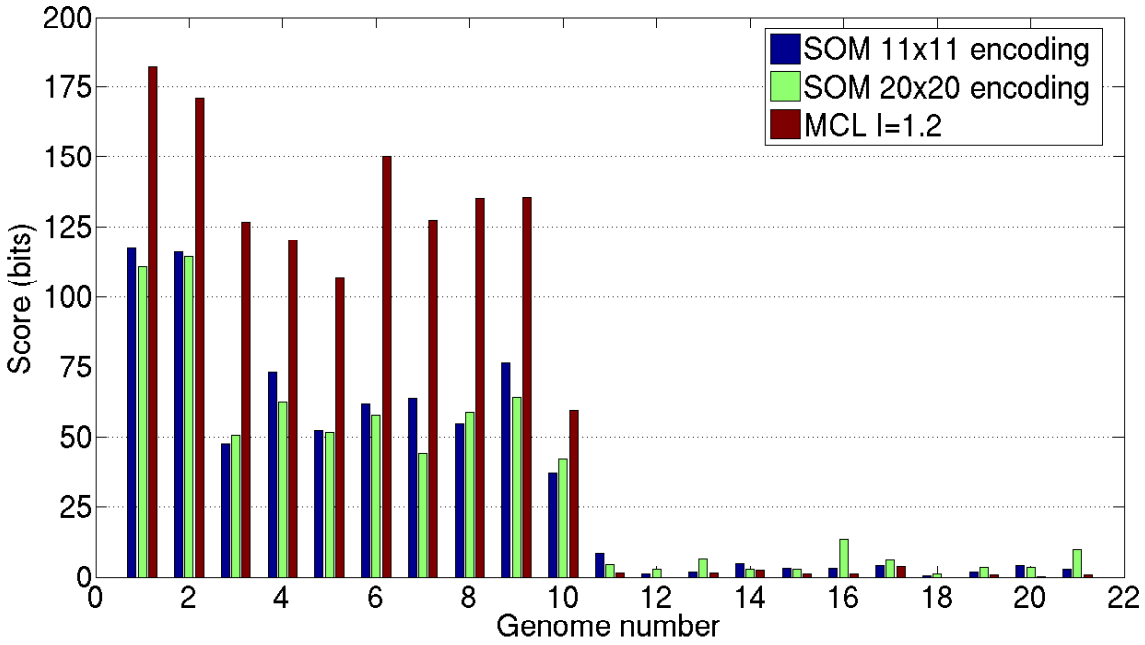

Figure 14. Clustering comparison: Average top hit score.

[6] In this case, it took hours to train a SOM whereas running BLAST and then MCL took minutes.

The genomes are listed in Appendix A.1. Genomes 1-10 are from *Herpesviridae* and the others are not.

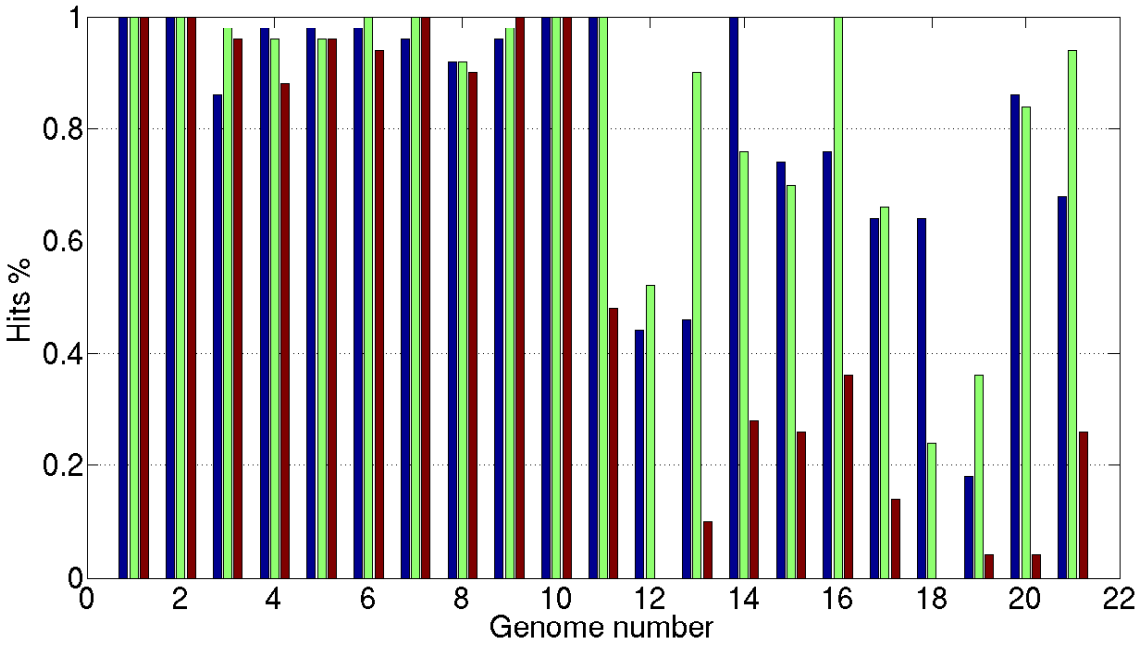

Figure 15. Clustering comparison: Percentage of fragments with hits.
The genomes are listed in Appendix A.1. Genomes 1-10 are from *Herpesviridae* and the others are not.
Blue bars represent the SOM using 11x11 encoding, green bars for the SOM using 20x20 encoding, and red bars for the results using MCL with $I = 1.2$.

5.4 Cross-validation of MCL Skeletons

A cross-validation test was conducted to determine the generalization power of the skeleton annotator for unknown fragments (i.e., in the face of fragments not from the genomes with which it was trained).

5.4.1 Experimental Design

A 3-fold cross-validation test was conducted on fragments of 700bp and 100bp originating from three virus families: *Herpesviridae*, *Bromoviridae*, and *Poxviridae*. The genomes of *Herpesviridae* are partitioned into three groups of approximately the same size. Pairs of

these groups are clustered using MCL and a skeleton is trained, resulting in three HMM skeletons each of which has not been trained with one of the three partitions. Each HMM skeleton is then tested with the fragments generated from the partition it was not trained with. This setup is likewise repeated with the other two families. The list of genomes and the partitioning is given in Appendix A.2.

5.4.2 Results

The cross-validation results against the fragments of length 700 bp will be discussed first. For all three families, the cross-validation skeletons gets hits from 90-100% of the fragments of most of the genomes, with only a couple getting a low 60-70%. The average top hit score of each genome's fragments is charted for each skeleton in Figures 16-18. Figure 16 shows the results for the 3-fold cross-validation test for *Herpesviridae*. Partition 1 contains genomes #1-14, partition 2 contains genomes #15-29, and partition 3 contains genomes #30-44. The results on the chart for partition 1 is from the HMMs trained from the genomes in partitions 2 and 3, the results for partition 2 is from the HMMs trained from the genomes in partitions 1 and 3, and finally the results for partition 3 is from the HMMs trained from the genomes in partitions 2 and 3. The chart shows that the top hits for genomes #14, 21, 26, and 27 of *Herpesviridae* score very low at below 10 bits. This occurred since these viruses are more unique than the others, with no other similar viruses having been part of the skeleton's training. In the presence of fragments from genomes of other families, the skeleton may not be able to classify these fragments correctly. However given the high hit percentage and average top hit scores for the majority of the *Herpesviridae* genomes, this skeleton detects fragments from family members outside the training set very well.

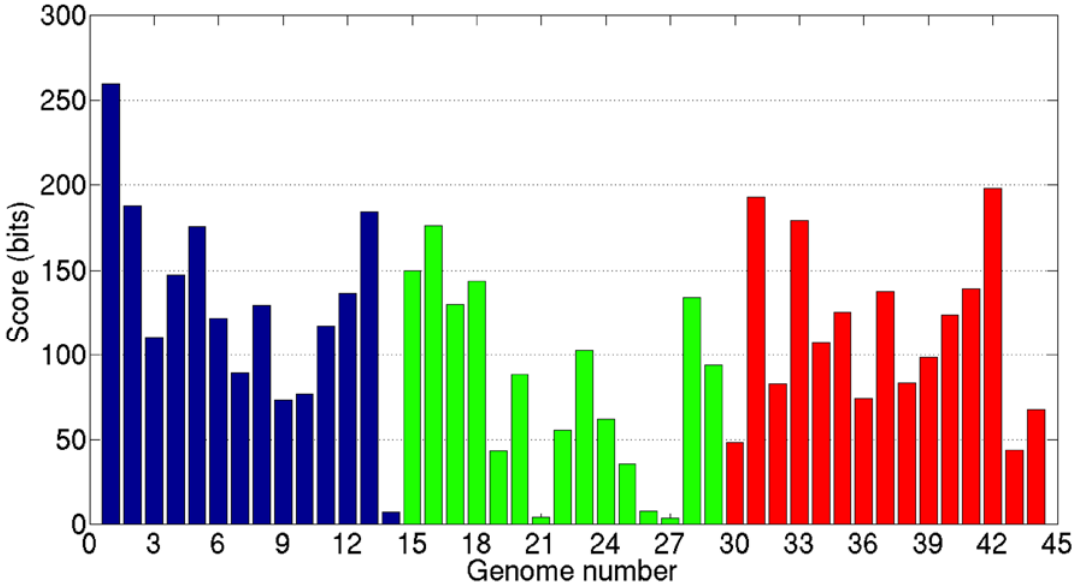

Figure 16. Herpesviridae 700 bp 3-fold cross-validation: Average top hit score.
Bar colors denote the three paritions.

The skeletons for *Bromoviridae* and *Poxviridae* performed even better than *Herpesviridae*'s skeleton as shown in Figures 17 and 18. For *Bromoviridae* the partitions are: genomes #1-7, #8-15, and #16-23, for partitions 1, 2, and 3, respectively. For *Poxviridae* the three partitions are: genomes #1-7, #8-14, and #15-22. Combined, these two skeletons achieve a low average top hit score of about 50 bits for only three genomes. Unlike *Herpesviridae* then, these skeletons can detect the low scoring genomes well. In short, these tests indicate that the HMM skeleton method can annotate 700 bp DNA fragments very well.

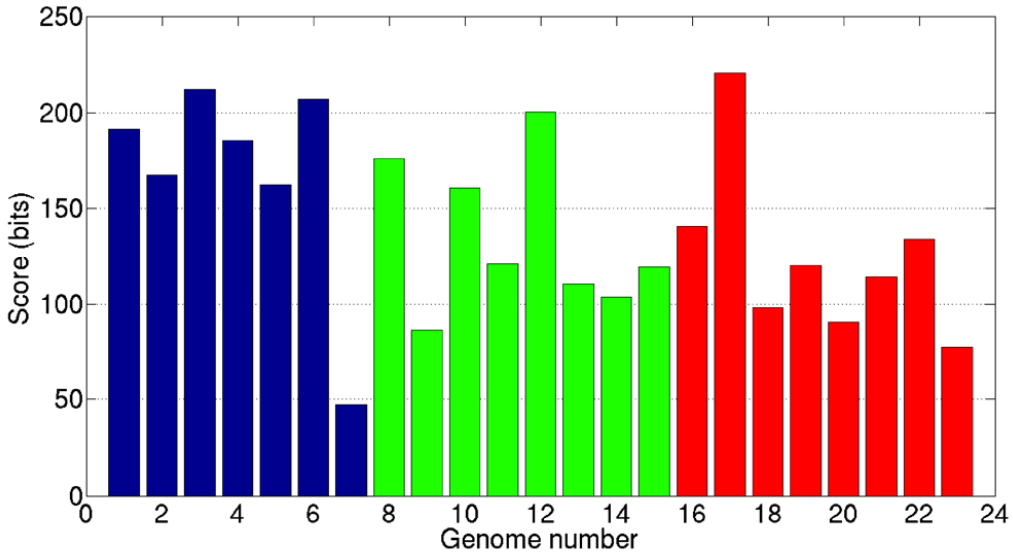

Figure 17. Bromoviridae 700 bp 3-fold cross-validation: Average top hit score.
Bar colors denote the three partitions.

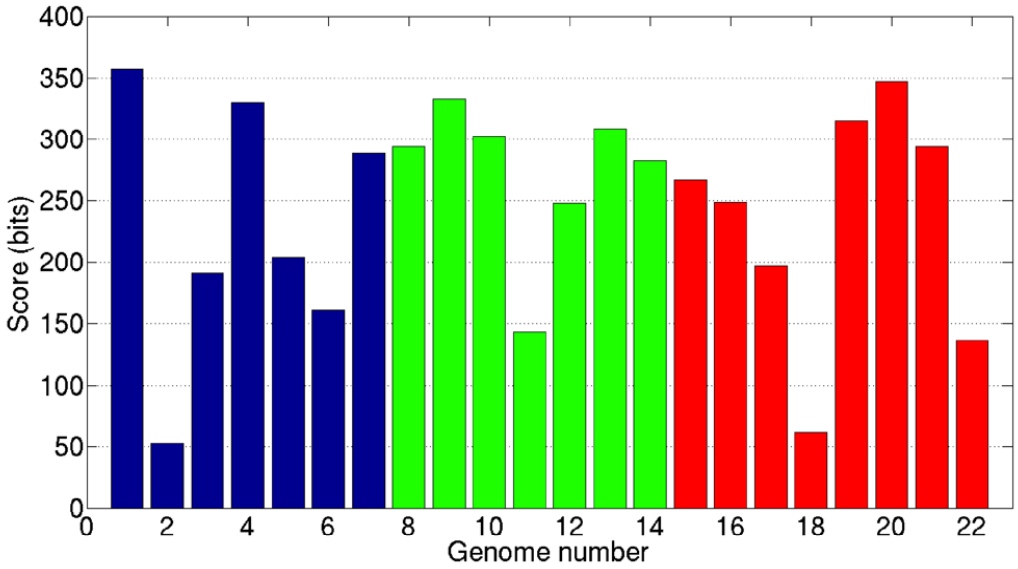

Figure 18. Poxviridae 700 bp 3-fold cross-validation: Average top hit score.
Bar colors denote the three partitions.

Next we analyze the results of the cross-validation tests for the *Herpesviridae* skeleton with 100 bp fragments. Figure 19 shows that, like the 700 bp case, fragments from genomes #14, 21, 26, and 27 score relatively low. Therefore the fragments from these genomes

cannot be classified with confidence, as was the case with 700 bp fragments. The scores overall are also lower compared to the 700 bp case, as these short fragments cannot achieve very long and high scoring matches.

Figure 20 shows the percentage of fragments of each genome in the 100 bp dataset that generated a hit against the Herpesviridae skeleton. While on average 94% of a genome's fragments got hits with the 700 bp dataset, only an average of 45% was obtained with 100 bp fragments with the remaining fragments being unclassifiable. Looking at all the fragments overall we see similar percentages with the skeleton detecting only 45% and 94% of the Herpesvirdae 100 bp and 700 bp fragments, respectively. Clearly, 700 bp fragments, which contain more information, are easier to detect and classify correctly. Bear in mind that the 45% achieved with 100 bp fragments is actually quite high. A preliminary study of a random subset of unassembled, 100 bp virus fragments from the Sargasso Sea using BLAST only achieved hits for 6% of the dataset.

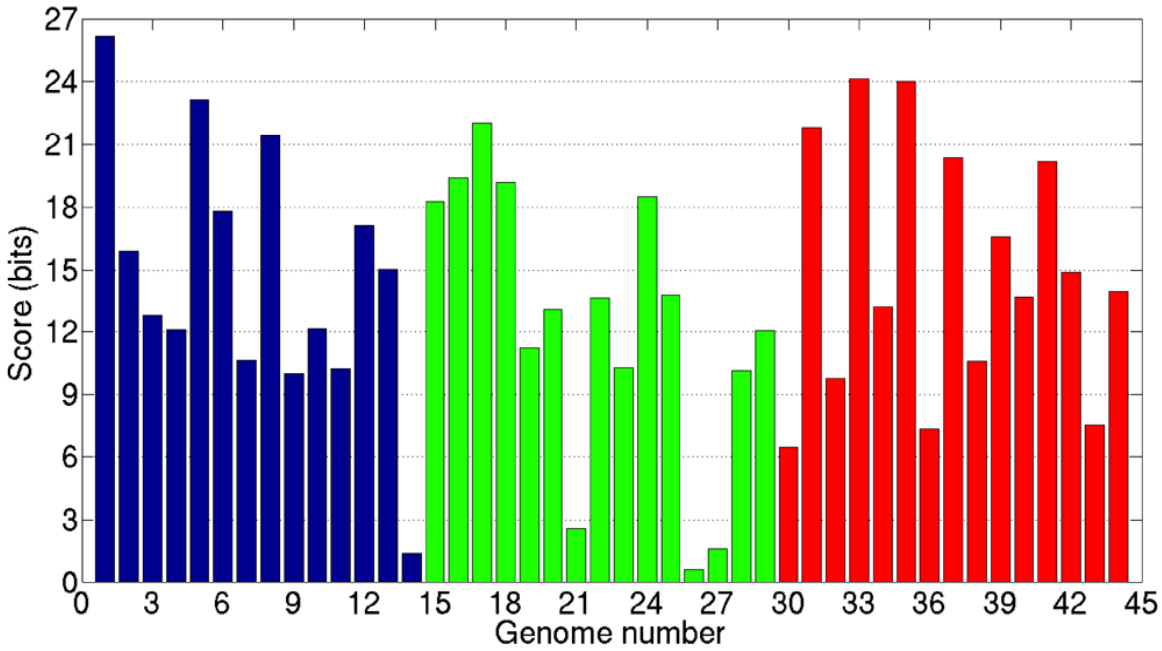

Figure 19. Herpesviridae 100 bp 3-fold cross-validation: Average top hit score.
Bar colors denote the three partitions.

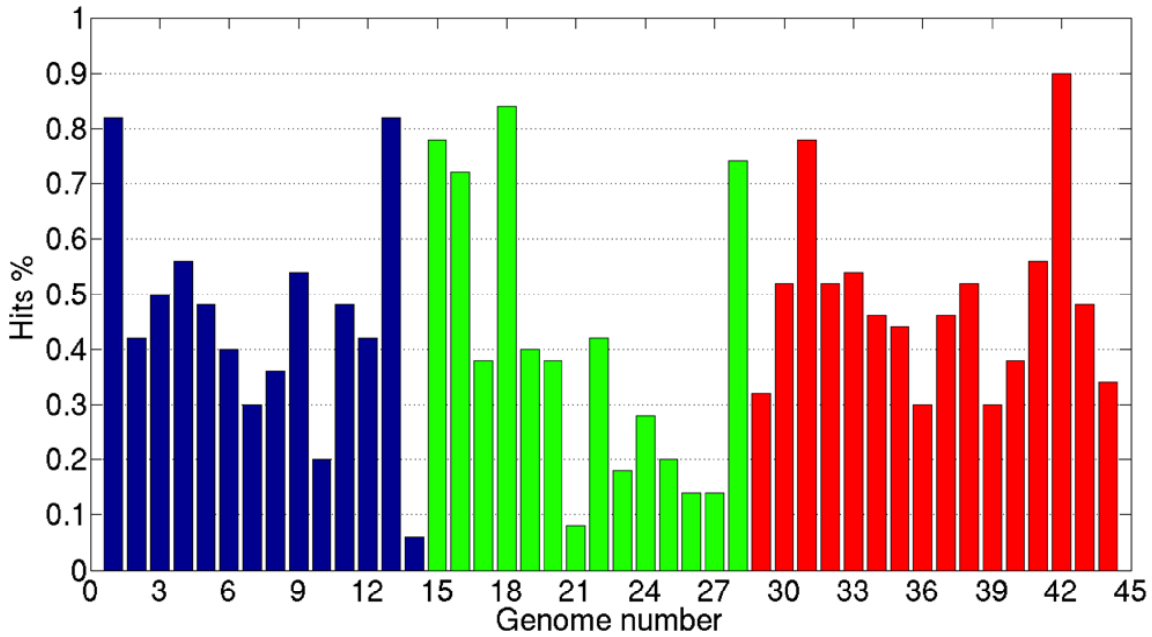

Figure 20. Herpesviridae 100 bp 3-fold cross-validation: Percentage of fragments with hits.
Bar colors denote the three partitions.

5.5 Multifamily Test

In this experiment, we build HMM skeletons for three viral skeletons and take fragments are from a range of families. The resulting skeletons are used to determine if the skeletons indeed recognize fragments from the family it represents and not fragments from others. Two sets of fragments were generated: one set of fragments with length 700 bp and the other set with length 100 bp.

5.5.1 Experimental Design

Unlike in the cross-validation test above, all genomes are used in the virus families to generate HMM skeletons for *Herpesviridae*, *Bromoviridae*, and *Poxviridae*. The test metagenome is generated by creating a total of 2350 fragments taken from the three virus families along with fragments from other viral genomes. The fragments were generated by random selection of 50 fragments from 47 different genomes: 12 genomes from *Herpesviridae*, 12 genomes from *Bromoviridae*, 12 genomes from *Poxviridae*, and 11 genomes from outside those families. This experiment tests the performance of the HMM skeletons against fragments that do not belong to the family it represents. The list of genomes contained in the fragment dataset is listed in Appendix A.3.

5.5.2 Results

First the results from the 700 bp fragment dataset are discussed. Figures 21-28 shows the average top hit score and hit percentage of each genome's fragments using the three family skeletons side-by-side for easy comparison. It is easily seen that each skeleton gets high average top scores on fragments from genomes they represent, and low scores from

fragments from genomes outside the skeleton's family. Figure 27 shows how low the top hit scores are for genomes outside of all three family skeletons. Figures 22, 24, 26, and 28 show that the genome hit percentages are a mixed bag. While for example the *Herpesviridae* skeleton catches most herpesvirus fragments, it also has a high rate of catching fragments from other families. But since these hits to outside family members score low on average, we can remedy the situation by setting an appropriate threshold score to reduce the number of false positives. Thus the results are desirable since we want each skeleton to only be sensitive to fragments originating from the taxon it represents.

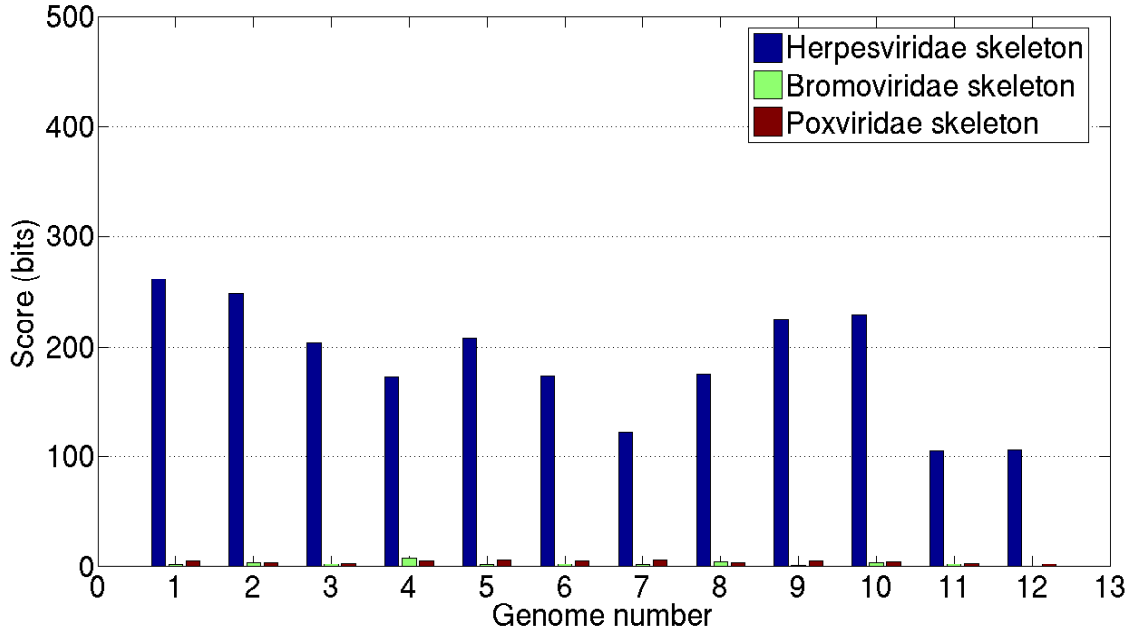

Figure 21. Skeleton comparison: Average top hit score for Herpesviridae genome fragments.

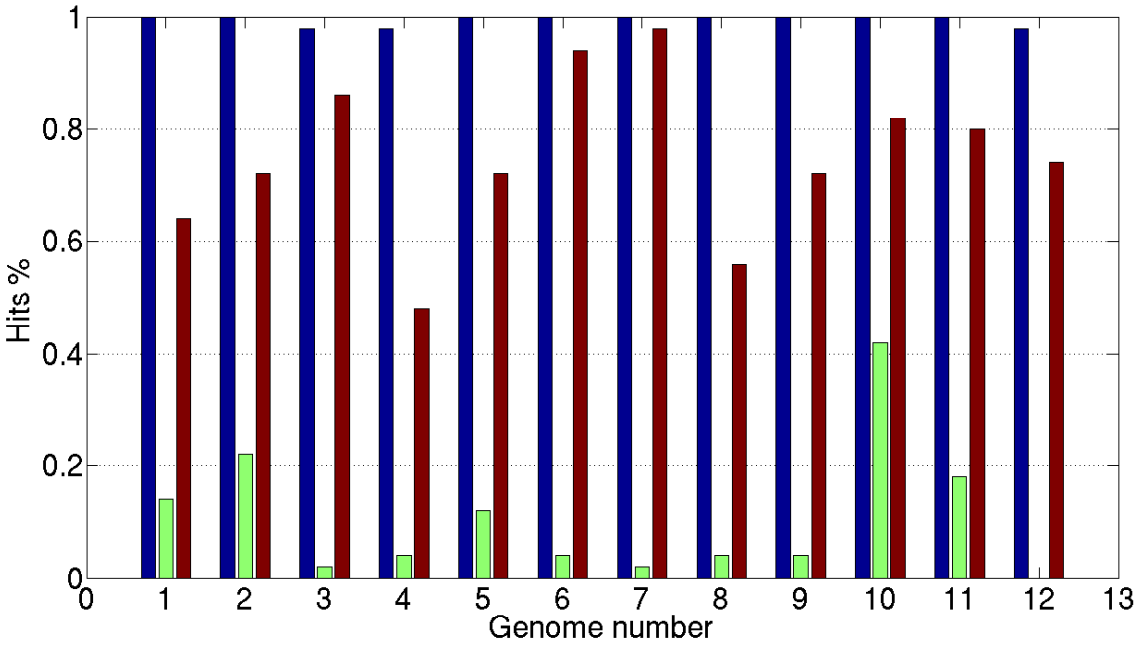

Figure 22. Skeleton comparison: Percentage of fragments with hits for Herpesviridae fragments.
Color legend: Blue: Herpesviridae skeleton, Green: Bromoviridae skeleton, Red: Poxviridae skeleton

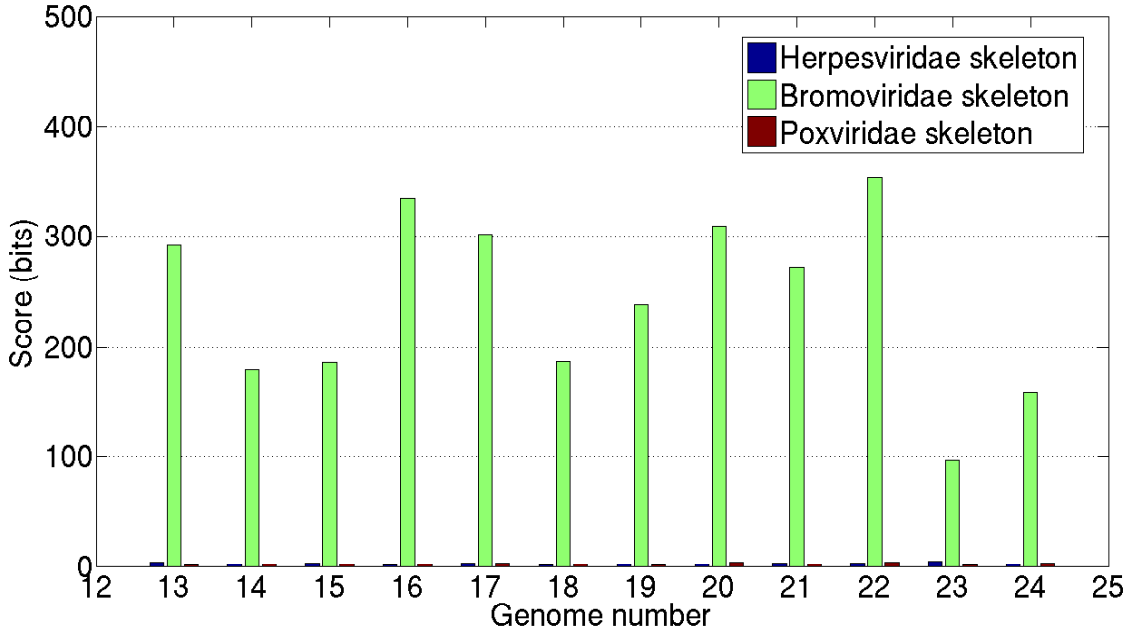

Figure 23. Skeleton comparison: Average top hit score for Bromoviridae genome fragements.

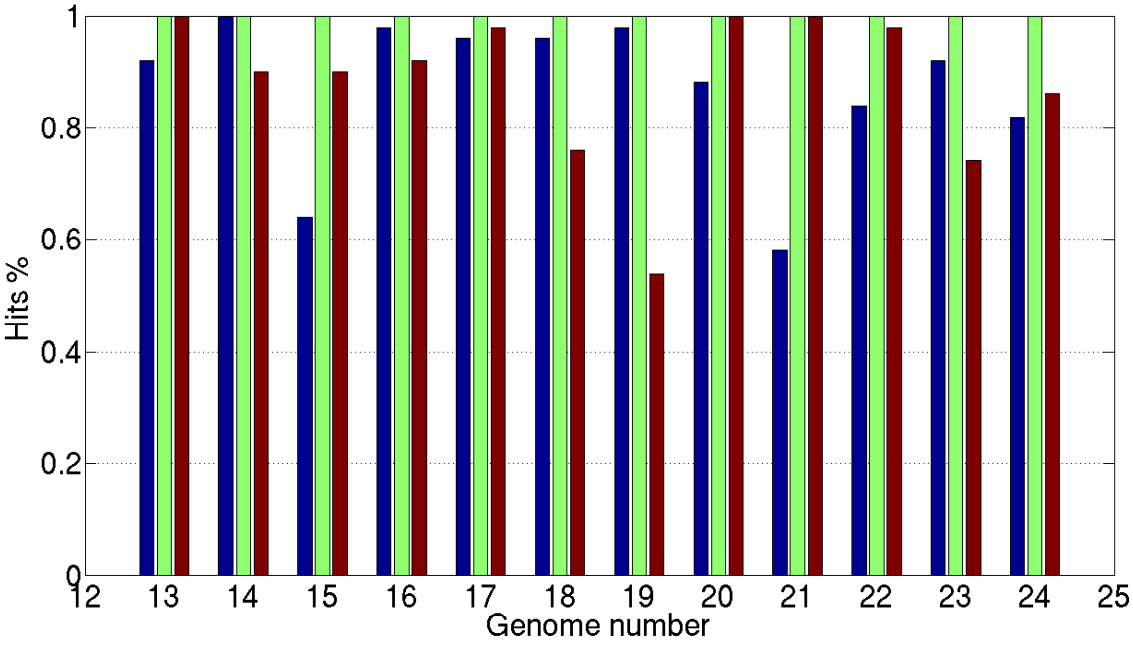

Figure 24. Skeleton comparison: Percentage of fragments with hits for Bromoviridae genome fragments. Color legend: Blue: Herpesviridae skeleton, Green: Bromoviridae skeleton, Red: Poxviridae skeleton

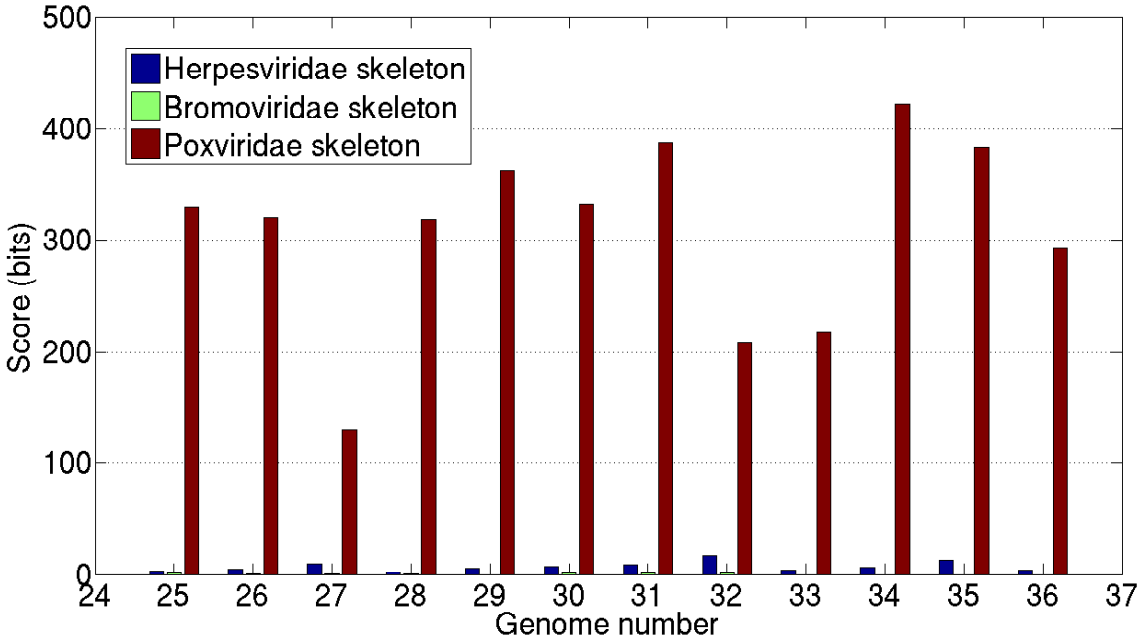

Figure 25. Skeleton comparison: Average top hit score for Poxviridae genome fragments.

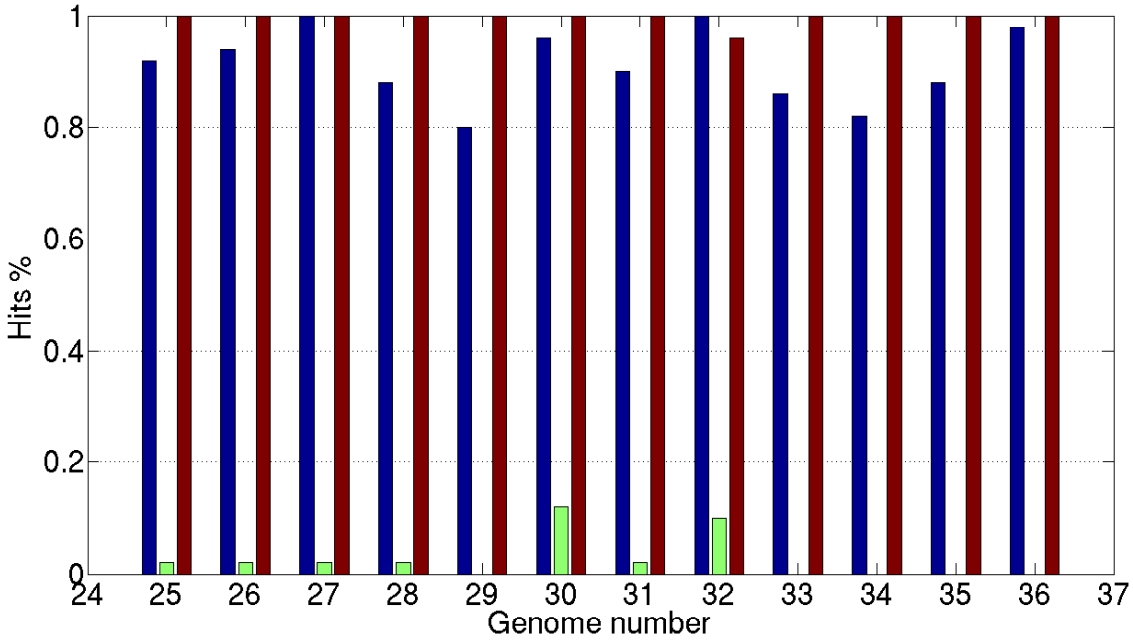

Figure 26. Skeleton comparison: Percentage of fragments with hits for Poxviridae genome fragments. Color legend: Blue: Herpesviridae skeleton, Green: Bromoviridae skeleton, Red: Poxviridae skeleton

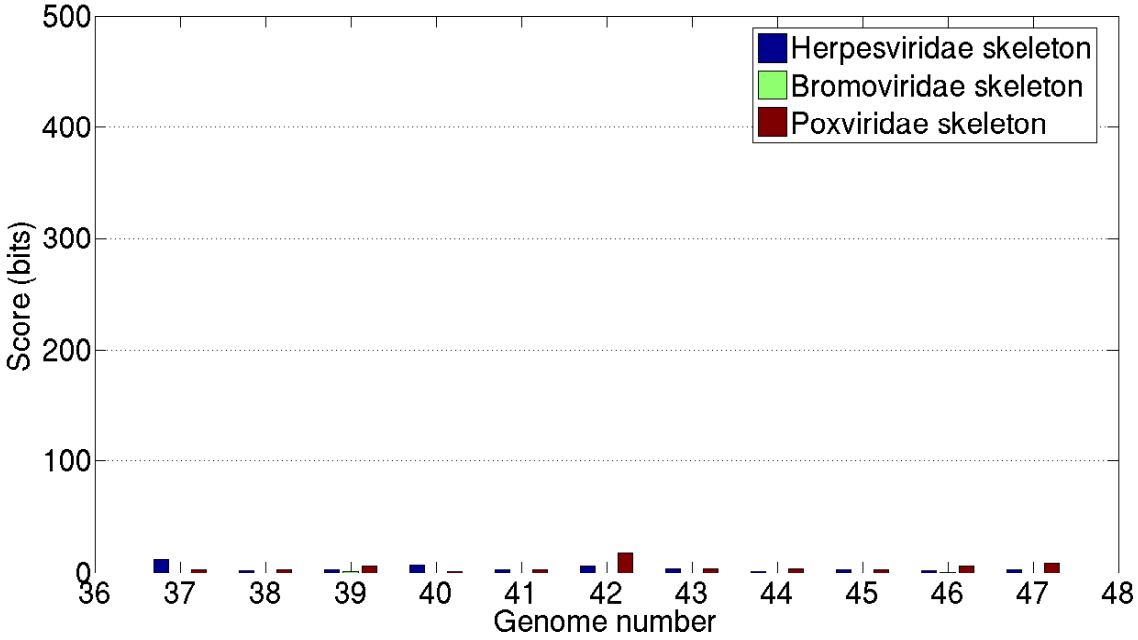

Figure 27. Skeleton comparison: Average top hit score for other genome fragments.

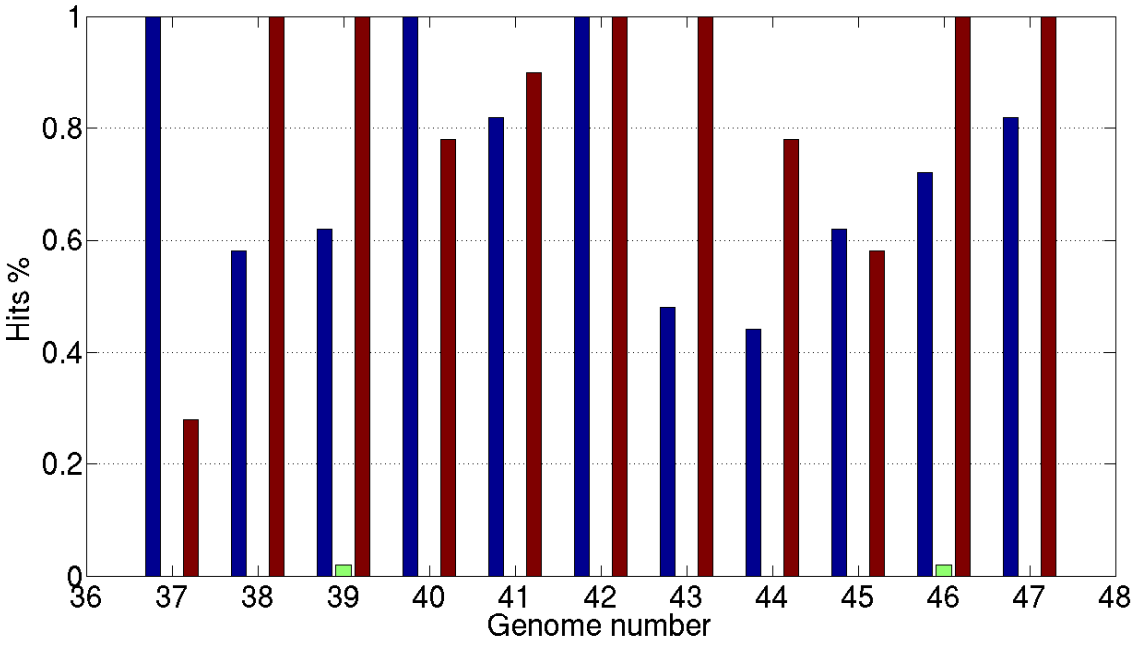

Figure 28 Skeleton comparison: Percentage of fragments with hits for other genome fragments.
Color legend: Blue: Herpesviridae skeleton, Green: Bromoviridae skeleton, Red: Poxviridae skeleton

Since we want to also target the pyrosequencing projects, we also query the 100 bp version of the dataset to the *Herpesviridae* skeleton. The results are illustrated in Figures 29 and 30. Compared to the 700 bp test, the scores and hit percentages have dropped. This is logical since the fragments are much shorter, making long high scoring sequence matches between the query sequence and a profile HMM impossible. Fewer fragments in this case can be classified with confidence, as will be further illustrated in the next section.

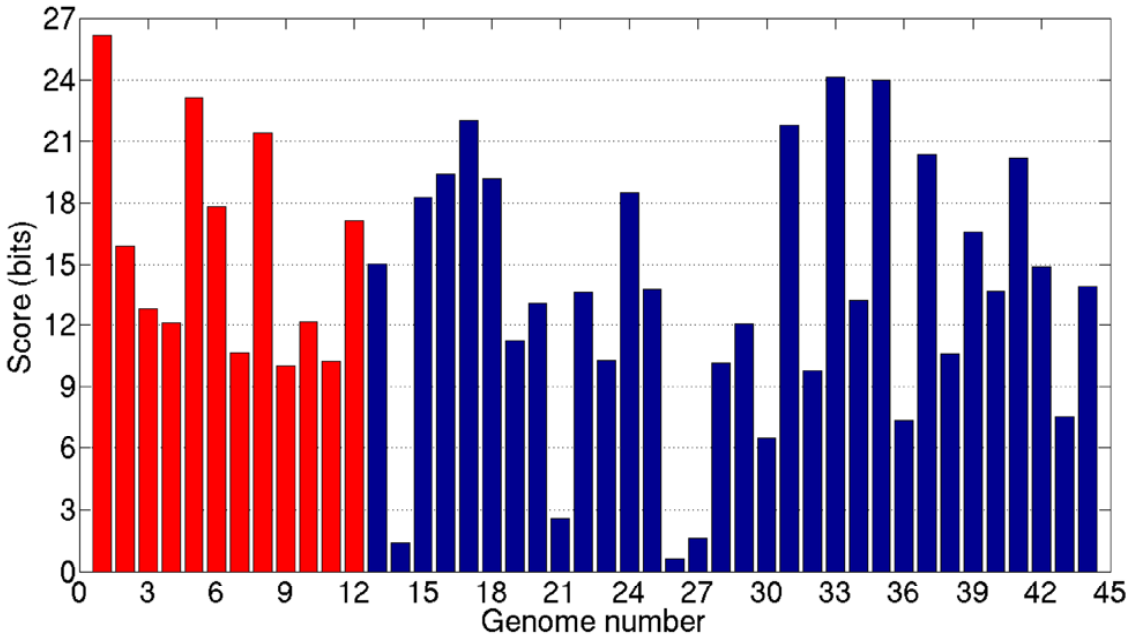

Figure 29. Herpesviridae skeleton: Average top hit score of 100 bp multifamily dataset. Red bars denote Herpesviridae genomes and the blue bars denote outside genomes.

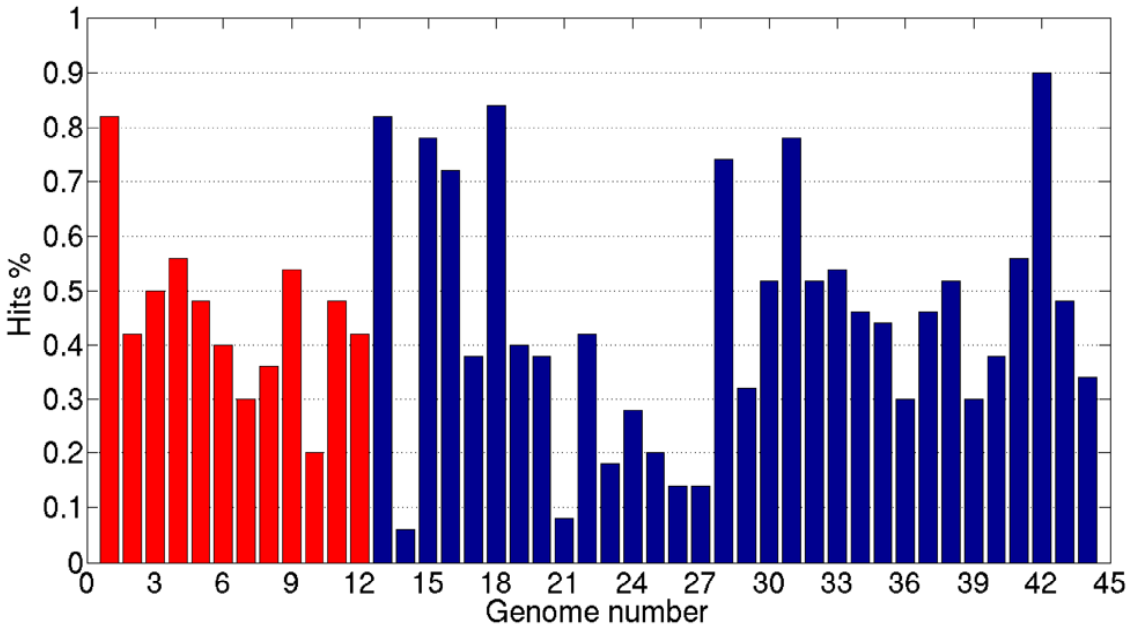

Figure 30. Herpesviridae: Percentage of fragments with hits for 100 bp multifamily dataset. Red bars denote Herpesviridae genomes and the blue bars denote outside genomes.

5.6 All Viral Taxa

In the previous section we undertook the analysis of three different families of viruses. But to validate the method, we need to evaluate the classification process for all the possible taxonomic classification of the viruses. In the following sections we will present the result of the analysis done on skeletons built from all possible viral taxa. For the selection of a good hit we divide this part of the experiment into two different threshold strategies: the single threshold and the multiple threshold strategies.

5.6.1 The Single Threshold Strategy: Experimental Design

A HMM skeleton is built from every known viral taxa using all the available viral sequence data. The taxonomic data is taken from each genome's NCBI[8] database lineage listing, which is based on the ICTV taxonomy. The taxa are then divided into nine lineage levels that do not correspond exactly to the ICTV ranks (Order, Family, Subfamily, etc.), as some viruses may be classified under more subclasses than others or may omit an ICTV rank. For example, Figure 7 shows the lineage of the species *Human herpesvirus 1*. The highest ranking taxon, Virus, is placed in level 0 in Anacle and the next ranking taxon, "dsDNA virus, no RNA stage," is placed into level 1, and so on with the genus Simplexvirus being placed into level 4. Some taxa are further divided into smaller-sublevels for up to four more levels. For levels 0-4, the training resulted in about 15,000 HMMs each. Levels 5 and above contain progressively less HMMs, as less and less genomes are classified up to these levels.

New 100 bp and 700 bp fragment datasets were created for this experiment. Each dataset consists of 5 fragments taken from 200 different genomes, for a total of 1000 fragments. The genomes were randomly selected and consist of 150 virus species and 50 non-virus species. The fragments are queried against all HMM skeletons. The fragments are then annotated or classified using the first approach described in Section 5.1.3, where the fragment is classified based on its top scoring HMM over all levels. To reduce false classifications we also introduce a threshold score. The top HMM must score above this threshold, or else we leave the fragment unannotated. Part of this experiment is to determine a threshold score that leads to good results.

---

[8] National Center for Biotechnology Information (NCBI): http://www.ncbi.nlm.nih.gov/

5.6.2 The Single Threshold Strategy: Results

In this section we leave a fragment unannotated or unclassified if the fragment has no hits or its top hit is below a certain threshold score. Otherwise we classify each fragment to the taxon containing the highest scoring HMM. Figure 31 and Table 3 shows the behavior of the system with threshold scores from 0 to 100 bits for the 100 bp dataset. For each threshold we count the number of virus fragments left unclassified, true positives (TP, virus fragments classified into a correct taxon), false negatives (FN, virus fragments classified into an incorrect taxon), false positives (FP, non-virus fragment classified into a viral taxon), and true negatives (TN, unclassified non-virus fragments). We then calculate the true positive rate (TPR) and false positive rate (FPR) as follows, $TPR = TP/(TP + FN)$ and $FPR = FP/(FP + TN)$. We plot the TPR (blue curve) and the fraction of unclassified virus fragments (green curve) versus the FPR. This is similar to the receiver operating characteristic (ROC) curve for binary classifiers, but here we also allow fragments to remain unclassified. The general trend is that as the threshold increases, our confidence in the predicted classifications also increases as the number of misclassifications drop. However the number of unannotated sequences also increases. So for example the experiment estimates that a threshold around 9 bits will classify about 8% of the non-virus fragments falsely, 1% of virus fragments falsely, 66% correctly, and leave 33% of virus fragments unannotated. More fragments can be annotated at the cost of having more false annotations.

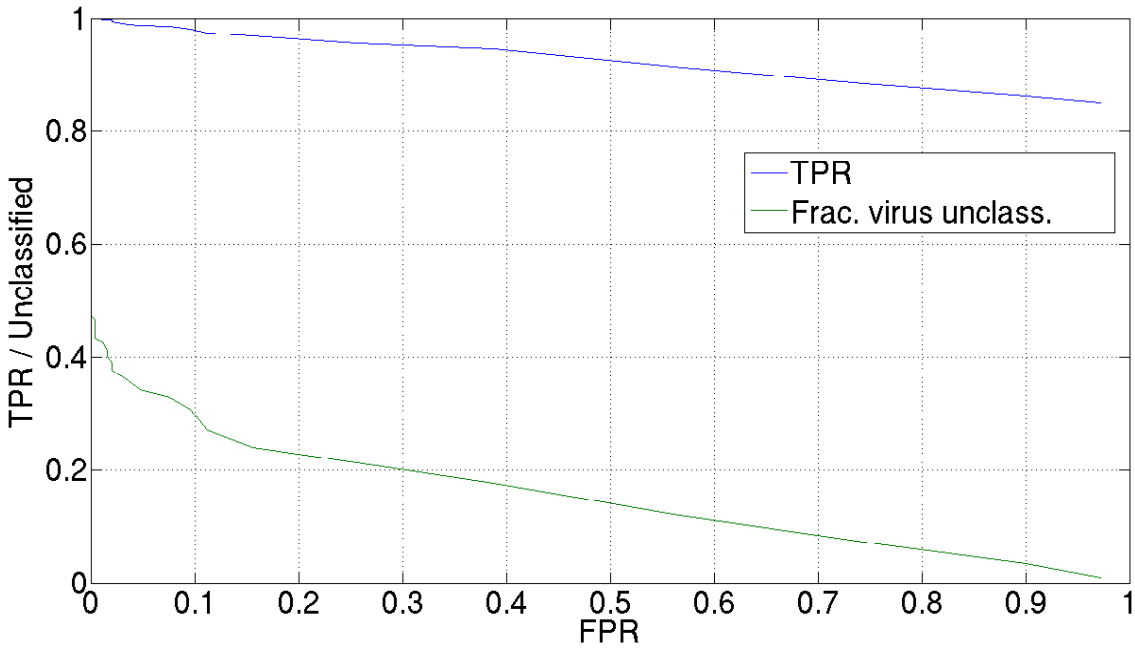

Figure 31. TPR/FPR curve for 100 bp dataset.

Table 3 also shows the trend of the number of fragments classified to a taxon in levels 0-3 at various threshold scores. Note that the number of fragments classified into levels 4 and above are 0, regardless of the threshold in this case. There is always a higher scoring HMM in a lower level. The trend shows that level 3 (approximately the taxa of the subfamily rank) obtains the highest counts than the lower, more general, levels. This is desirable, as we would prefer a more specific classification to a more general one. Of course taxa in levels 4 and above give more information than those of level 3, as they are even more specific, lower ranking taxa. So it would be even more desirable to obtain classifications in even higher levels. To do this, we try a classification approach that works from the bottom-up, from the lower ranking taxa to higher ranking taxa, as will be described in Section 5.6.3.

Threshold (bits)	TPR	FPR	Unclassified (%)	Lvl 0		Lvl 1		Lvl 2		Lvl 3	
				F	T	F	T	F	T	F	T
0	.851	.972	.933	0	175	21	107	34	103	56	247
5	.956	.256	21.3	0	133	6	99	11	100	9	232
9	.986	.076	32.8	0	119	1	89	1	94	5	195
15	.998	.016	41.2	0	108	0	77	1	83	0	172
20	1.00	0.00	47.3	0	99	0	70	0	73	0	153

Table 3. Distribution of classified virus fragments for 100 bp dataset.
For a selection of threshold scores, the number of true (T) and false (F) classifications is shown for each lineage level.

Figure 32 and Table 4 for the 700 bp dataset are analogous to Figure 31 and Table 3. The data was generated by testing threshold scores from 0 to 100 bits in increments of 1. While the graph shows the same trends as the 100 bp case, clearly the results here are superior since the number of unclassified virus fragments have dropped significantly. Figure 32 indicates that we can lower the FPR to 2.0 with not much of an increase in unclassified fragments. There is however a sharp increase in the number of unclassified fragments if one tries to lower the FPR further. The trend of the distributions of the fragments classified at the various levels is similar, where level 3 again obtains the most classifications. With the higher number of classified virus fragments and high TPR in general, this is clear evidence that the traditional sequencing methods are superior to the cheaper Pyrosequencing that generates the smaller 100 bp fragments in terms of obtaining quality annotation.

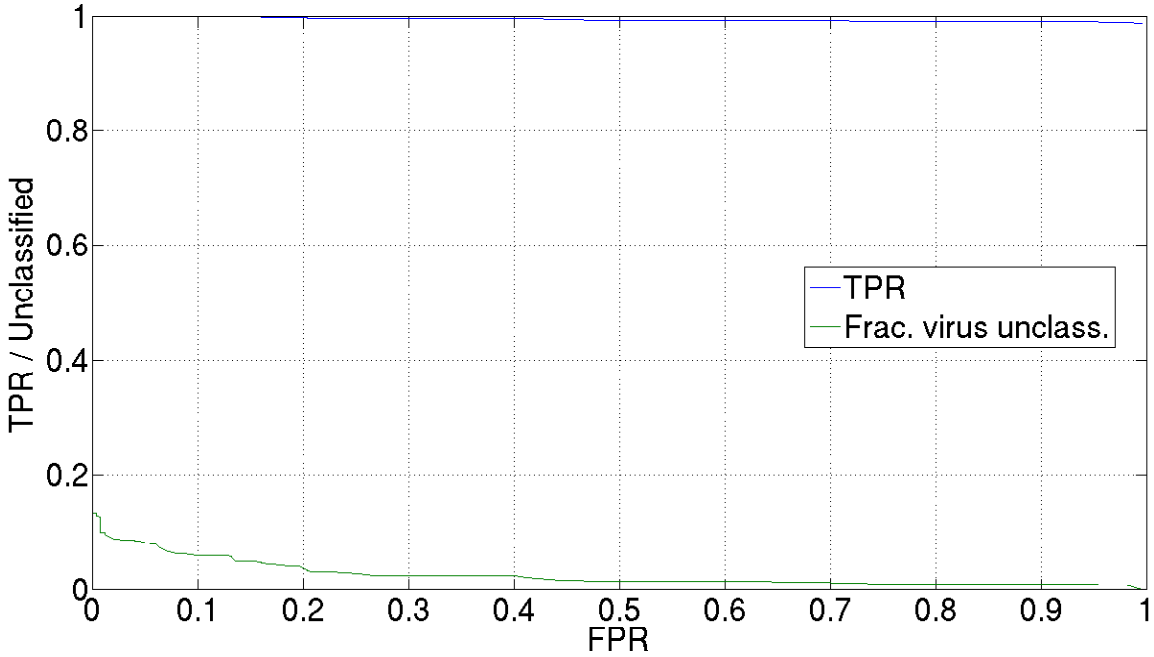

Figure 32. TPR/FPR curve for 700 bp dataset.

Threshold (bits)	TPR	FPR	Unclassified (%)	Lvl 0		Lvl 1		Lvl 2		Lvl 3	
				F	T	F	T	F	T	F	T
0	.988	.996	0	0	160	2	110	3	135	4	336
5	.991	.928	.933	0	155	1	110	2	135	4	336
10	.993	.528	1.47	0	154	0	109	2	135	3	336
15	.996	.356	2.40	0	153	0	107	1	134	2	335
20	.996	.252	2.80	0	153	0	106	1	133	2	334
25	.996	.204	3.20	0	153	0	105	1	132	2	333
30	.997	.180	4.27	0	153	0	105	1	130	1	328

Table 4. Distribution of classified virus fragments for 700 bp dataset.
For a selection of threshold scores, the number of true (T) and false (F) classifications is shown for each lineage level.

5.6.3 The Multiple Threshold Strategy: Experimental Design

In the previous section, we saw that annotating a fragment based on its top hit overall resulted in no annotations in the more specific taxa in levels 4 and above. However, it is desirable for fragments to be annotated as specifically as possible. In this section, we take a different approach to annotating the fragments that will remedy this situation.

The same 100 bp and 700 bp fragment datasets and the HMM skeletons of all the viral taxa of Section 5.6.1 are used here. However here we use the second approach to annotating a fragment described in Section 5.1.3. In this approach we classify a fragment to the lowest ranking taxon that contains a hit above a certain threshold score. That is, we first query a fragment to all taxa of the lowest rank, level 8 in this case. If the top hit at this level is above a certain threshold score, we classify the fragment based on that hit and we no longer need to query the fragment to further levels. Otherwise the fragment is left unclassified and is queried to all the taxa in the next rank up, level 7, where we repeat the process. This approach allows a fragment to be annotated to a more specific taxon despite getting a higher scoring hit with a cluster of a more general taxon. This approach also saves computing time, as a fragment does not have to be queried against every taxa skeleton.

In general, we can have a different threshold score for each level. So unlike for the first method, we need multiple threshold scores, one for each level. We try to estimate good threshold scores by producing ROC-like TPR/FPR curves for each level. We first use the entire fragment dataset to query the level 8 taxa and produce the level 8 TPR/FPR curve. Any fragments classified at this level are then removed from the dataset. The resulting smaller dataset is used to query against the level 7 taxa and to produce the level 7 TPR/FPR curve. We then shrink the dataset again by removing the classified fragments, and query against the next level, as so on.

5.6.4 The Multiple Threshold Strategy: Results

The results of the 100 bp dataset will be discussed first. Figures 33-39 give the TPR/FPR curves for levels 6 to 0. Only a tiny fraction, 10 fragments, of our dataset have lineages that

go all the way down to levels 7 and 8. The TPR/FPR curves for these levels are then not informative, and have been omitted. The threshold scores for these two levels were set low enough to classify the 10 fragments to a level 8 taxon.

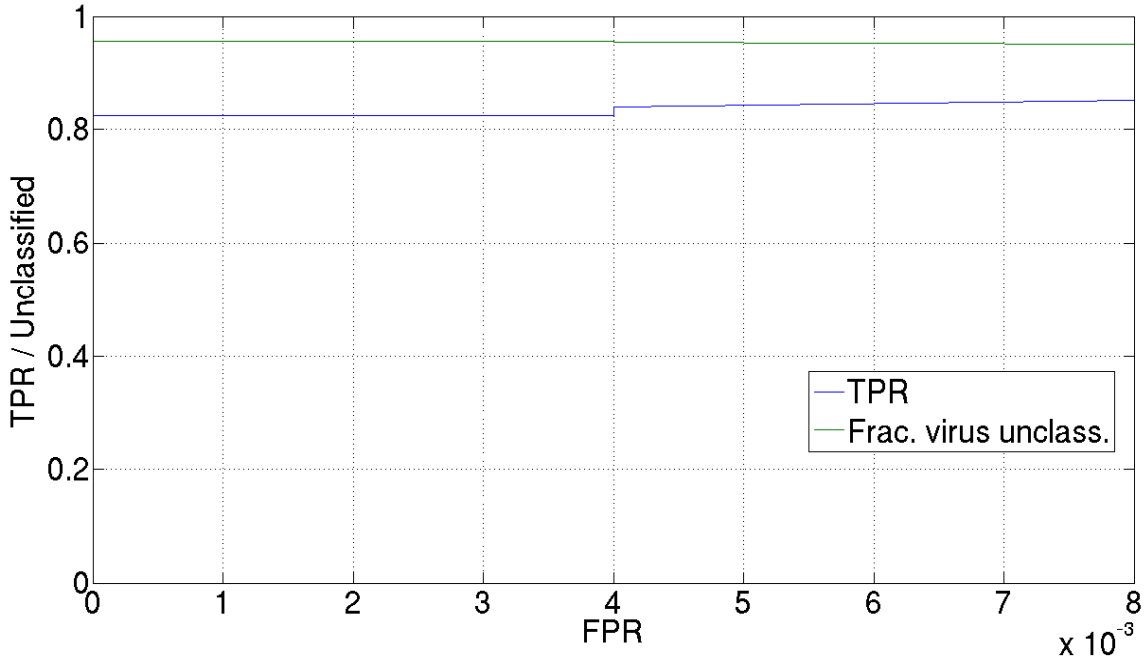

Figure 33. 100 bp: TPR/FPR curve for level 6.

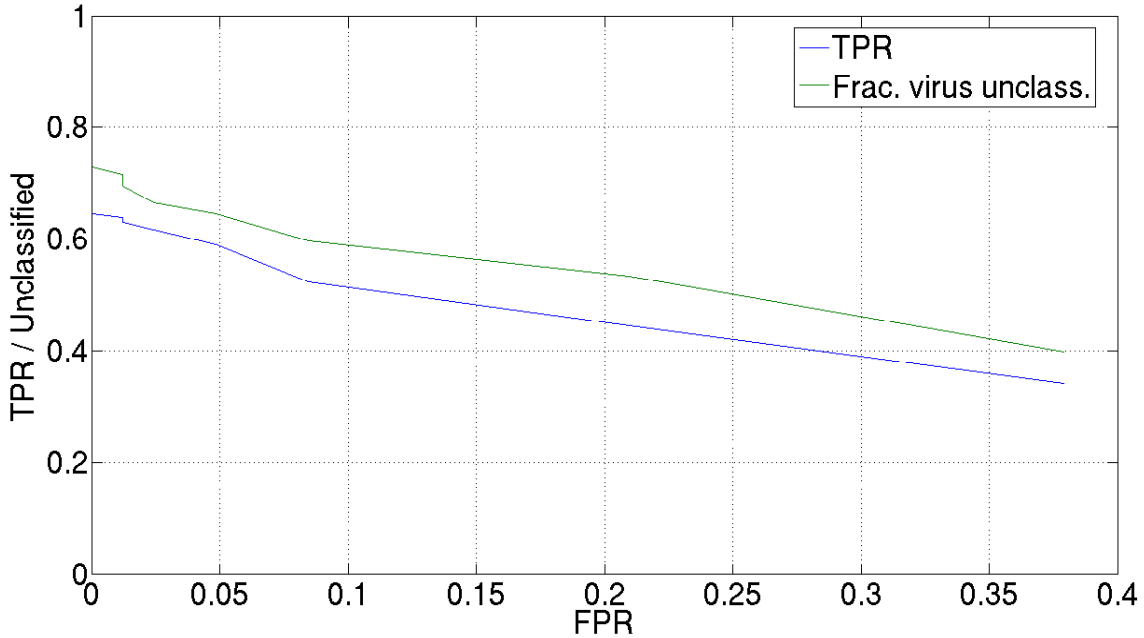

Figure 34. 100 bp: TPR/FPR curve for level 5.

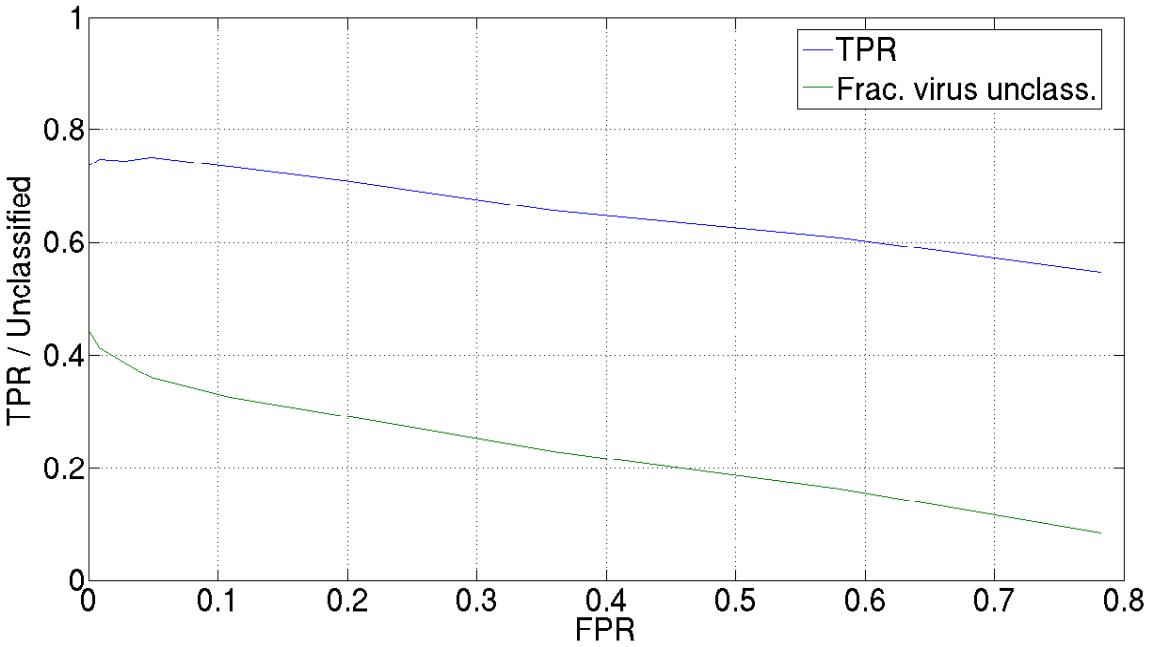

Figure 35. 100 bp: TPR/FPR curve for level 4.

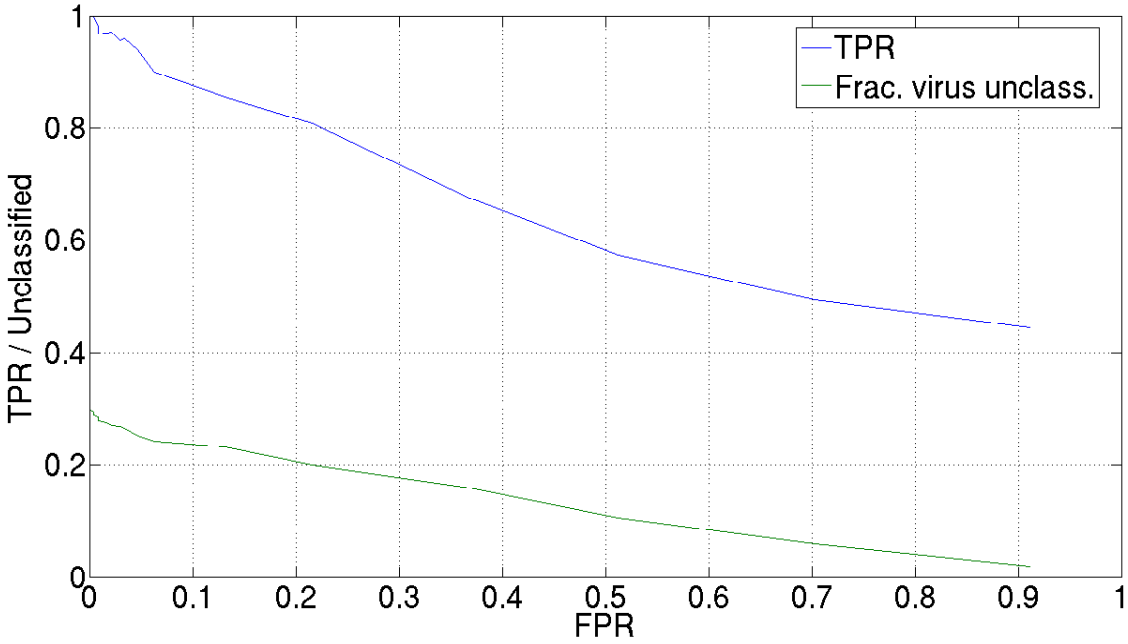

Figure 36. 100 bp: TPR/FPR curve for level 3.

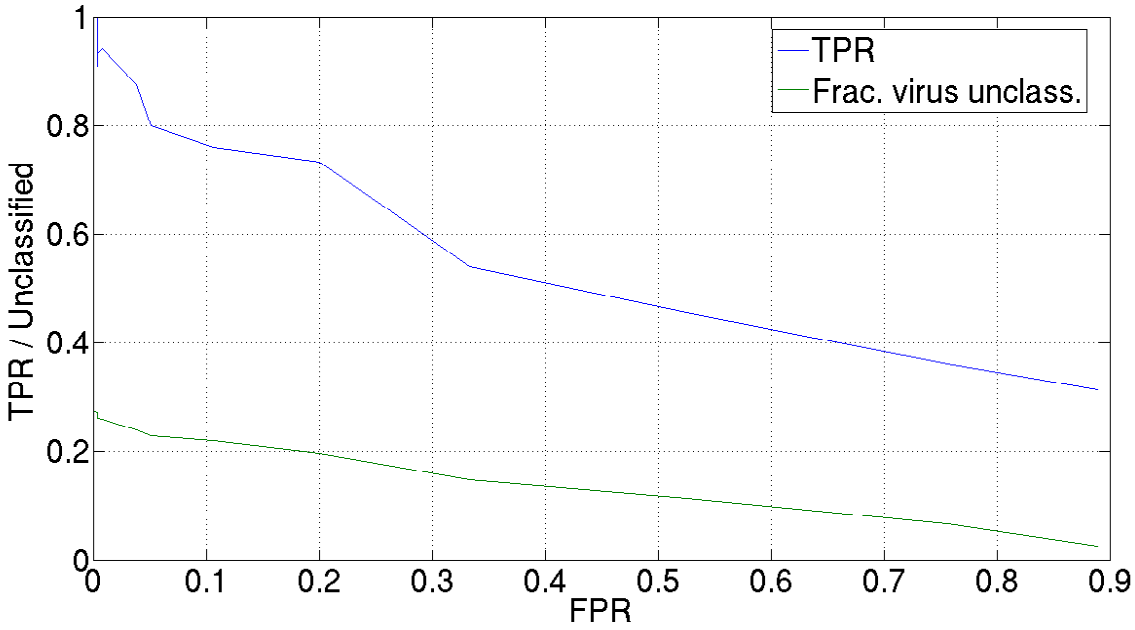

Figure 37. 100 bp: TPR/FPR curve for level 2.

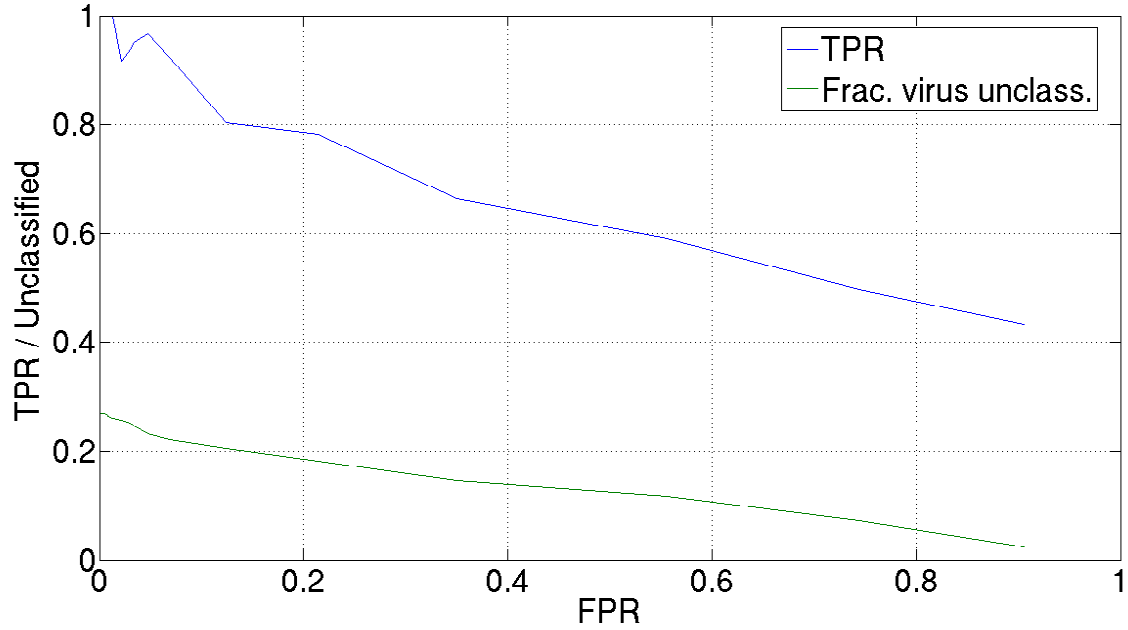

Figure 38. 100 bp: TPR/FPR curve for level 1.

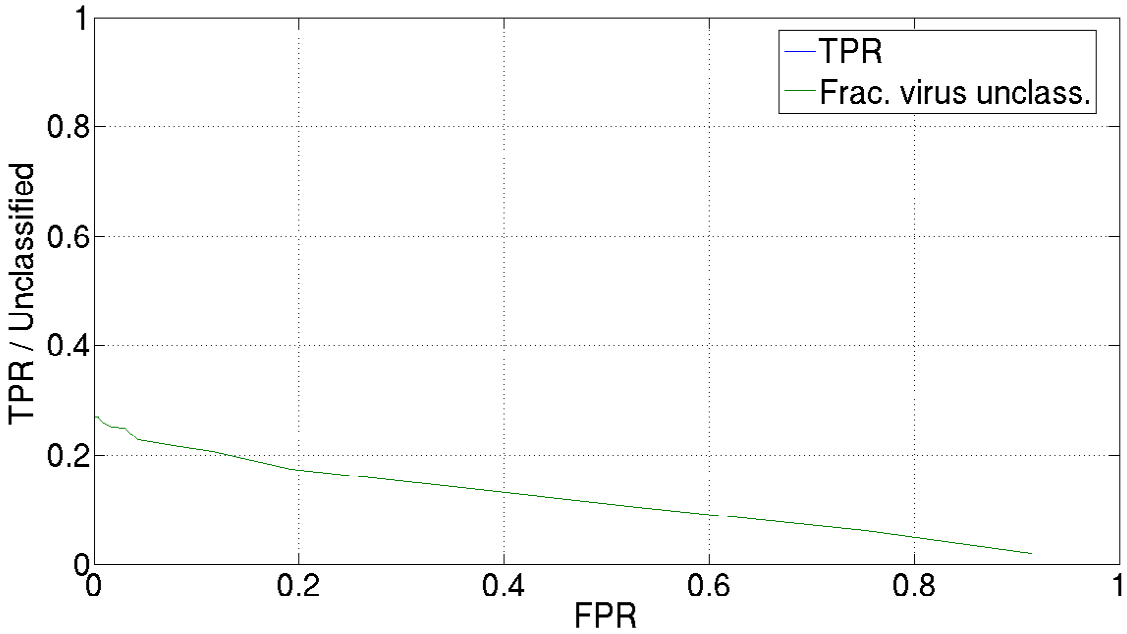

Figure 39. 100 bp: TPR/FPR curve for level 0.
In this case TPR is always 1 since here is only one class, Virus.

The plots show that as we lower the threshold score, the TPR decreases and the FPR increases, similar to what we saw in the Section 5.6.2. This occurs because with a lower threshold score, we classify more fragments. Of course, the more non-virus fragments that are successfully annotated, the higher the FPR. We are also classifying more virus fragments, but we are accepting more FNs than TPs, resulting in a lower TPR. According to the TPR/FPR curves, the peak TPR for levels 1-6 occur at a low FPR, and so these points are good threshold candidates. Since level 0 only consists of the taxon Virus, the TPR is always 1, and we need to pick a threshold score that obtains a desirable FPR and number of unannotated fragments. For levels 1-6 we selected the threshold score that gave the peak TPR and for level 0 we picked the threshold that gave about 0.01 FPR. We then calculated the overall TPR and FPR based on all the classifications made across all levels, and

compared it to the to result with similar FPR of the single threshold approach in the last section. This is summarized in Table 5.

Method	TPR	FPR	Unclass. (%)
SING.	.986	.076	32.8
MULT.	.782	.072	26.13

Method	Lvl 0		Lvl 1		Lvl 2		Lvl 3		Lvl 4		Lvl 5		Lvl 6		Lvl 7		Lvl 8	
	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T
SING.	0	119	1	89	1	94	5	195	0	0	0	0	0	0	0	0	0	0
MULT.	0	7	0	2	0	6	1	58	86	261	30	66	4	23	0	0	0	10

Table 5. 100 bp: Single threshold vs multiple thresholds.

From Table 5 we can see that while the single threshold approach results in a higher TPR at a similar FPR, the multiple threshold approach resulted in less unannotated fragments and many annotations at level 4 (representing approximately the taxa of the genus rank) and above, which the single threshold approach does not obtain at all. In fact for this multiple threshold approach, level 4 achieves the highest annotation count. Again this is very desirable, as it is more informative to have annotation as specific as possible. This however does come at the cost of more false classifications. It is up to users to decide which approach is more suitable for their research.

The results of the 700 bp show the same trends as the 100 bp, as can be seen in Figures 40-46. Again we picked threshold scores that had the peak TPR for levels 1-8, and picked the threshold that corresponds to roughly 0.01 FPR. We also again calculate the overall TPR and FPR of the classifications and compare them to the single threshold results of similar FPR, which is shown in Table 6. We again see that the single threshold is superior in terms of having fewer false classifications with its low TPR, but the multiple threshold approach again obtains more specific classifications. Note that in comparison to the 100 kb case where level 4 earned the most hits, our results show the 700 kb earned the most in level 3. If we

lower the threshold for level 4, however, we can boost the number of fragments classified at this level at the cost of a lower TPR.

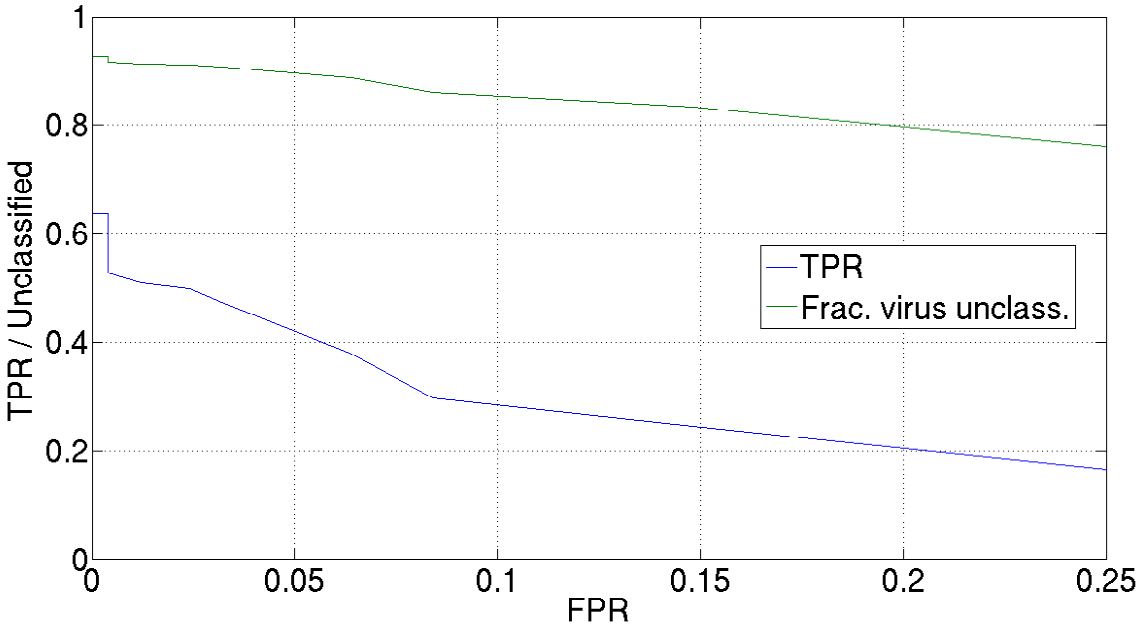

Figure 40. 700 bp: TPR/FPR curve for level 6.

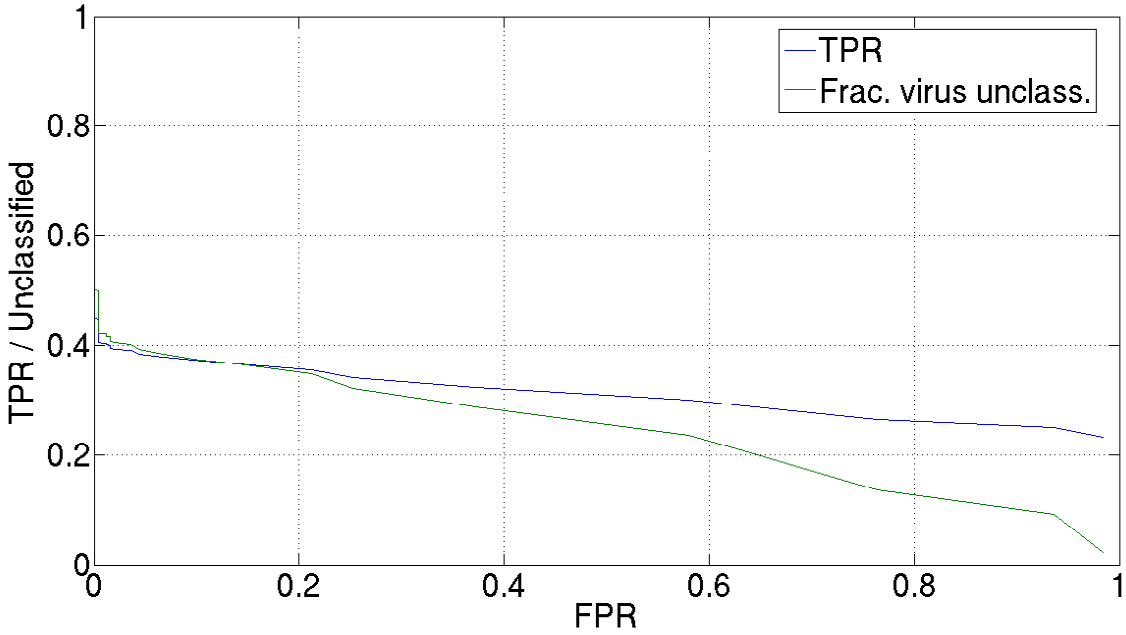

Figure 41. 700 bp: TPR/FPR curve for level 5.

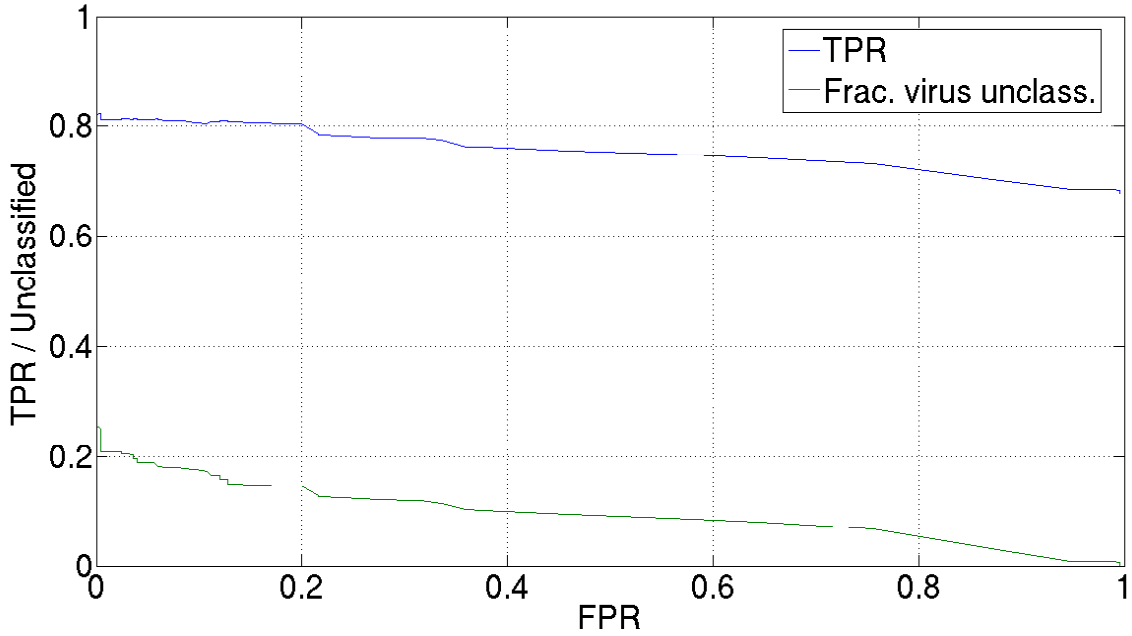

Figure 42. 700 bp: TPR/FPR curve for level 4.

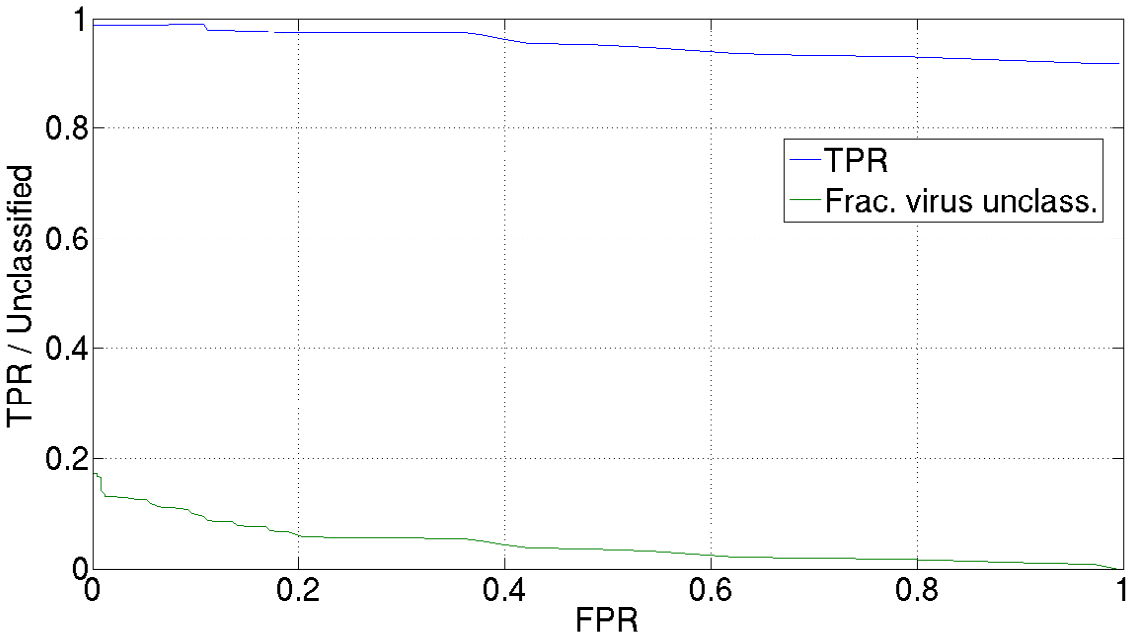

Figure 43. 700 bp: TPR/FPR curve for level 3.

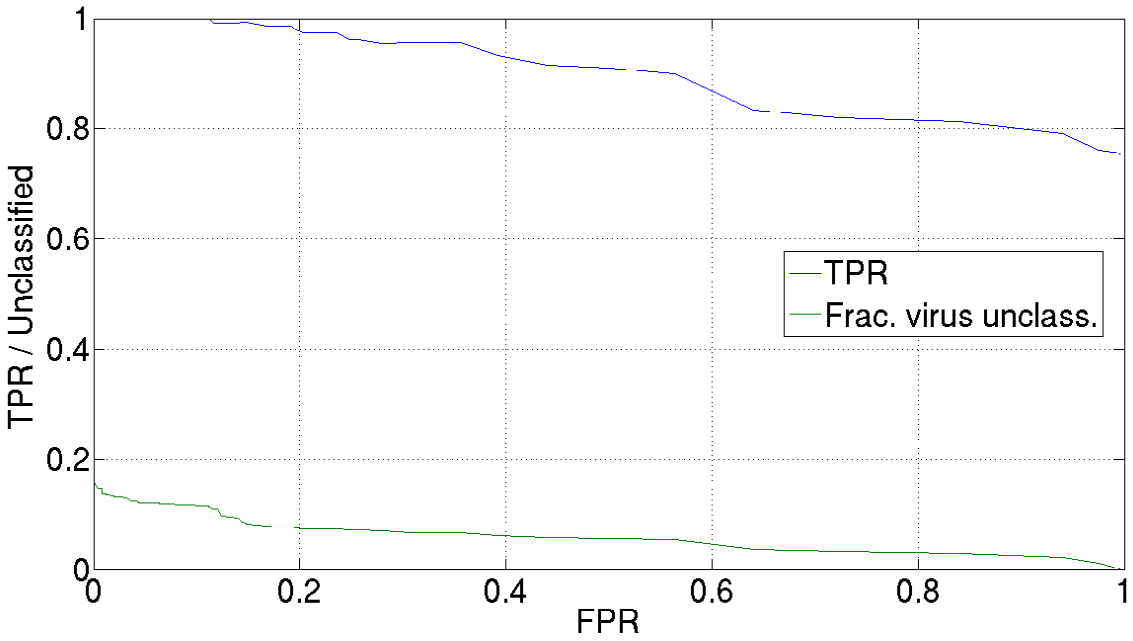

Figure 44. 700 bp: TPR/FPR curve for level 2.

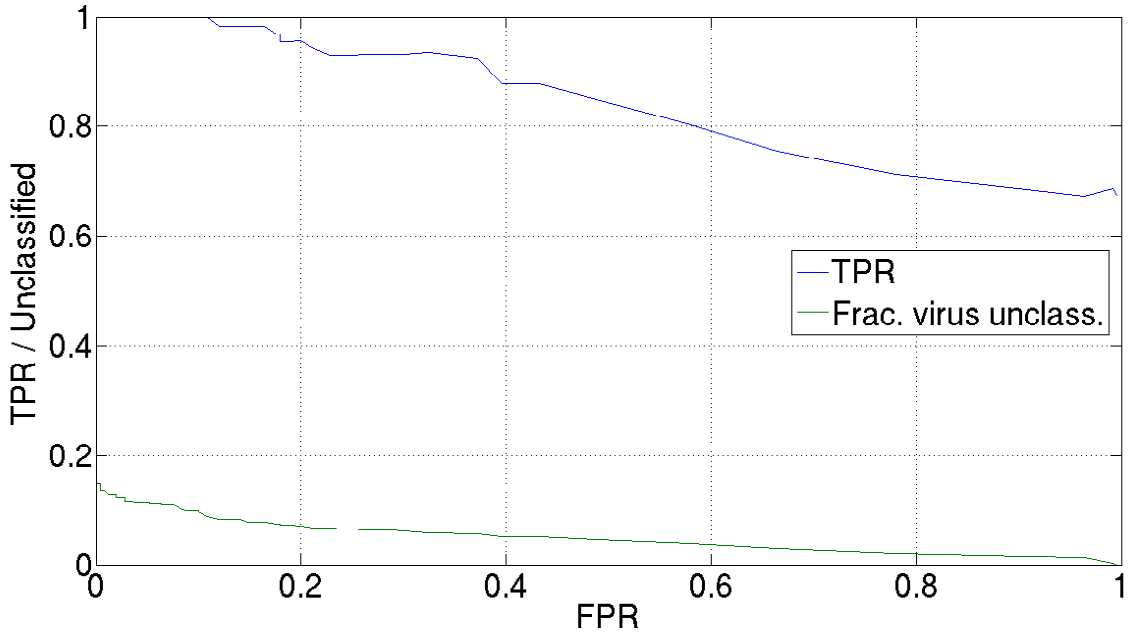

Figure 45. 700 bp: TPR/FPR curve for level 1.

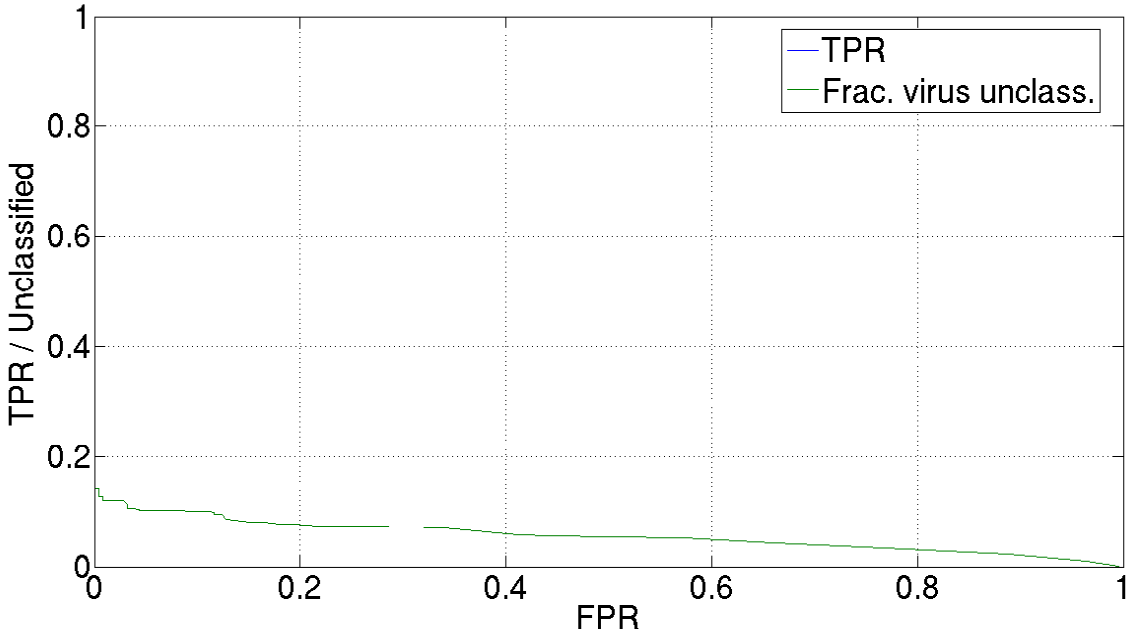

Figure 46. 700 bp: TPR/FPR curve for level 0.
Note that the TPR is always 1.

Method	TPR	FPR	Unclass. (%)
SING.	1.00	.008	10.0
MULT.	.970	.008	11.9

Method	Lvl 0		Lvl 1		Lvl 2		Lvl 3		Lvl 4		Lvl 5		Lvl 6		Lvl 7		Lvl 8	
	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T
SING.	0	147	0	94	0	129	0	305	0	0	0	0	0	0	0	0	0	0
MULT.	0	24	0	7	0	89	0	339	12	138	6	12	2	22	0	0	0	10

Table 6. 700 bp: Single threshold vs multiple thresholds.

Chapter 6: Discussion

The results of the experiments comparing MCL and SOM clustering show that MCL is superior to SOMs in running time performance and in the annotation performance of the resulting skeletons. We therefore chose to use MCL exclusively in our experiments. The cross-validation experiments on three virus families indicated that Anacle has great generalization since it can correctly annotate fragments from genomes not present in the training set. We then showed that Anacle achieves good performance in taxonomic annotation using experiments through a small multiple family test and through a large test against all viral taxa. For example, we were able to classify 67.2% of our artificial metagenome consisting of small unassembled 100 kb fragments with 0.986 TPR and a low FPR of 0.076. We also showed that we can trade-off among the TPR, FPR, the percentage of fragments left unclassified, and how specific the classifications are by tuning the threshold scores. We saw that annotating a fragment based on its top scoring HMM over a single threshold score resulted in high TPR and low FPR, but at the cost of having the classifications being in more general taxa (subfamily rank and above). On the other hand we saw that when using a multiple threshold scores, one for each level of taxonomy, we can obtain more specific classifications at the genus rank and below at the cost of having a lower TPR.

Therefore Anacle is capable of giving quality annotation to short, unassembled fragments, unlike other methods, like PhyloPythia, that require longer sequences or contigs that would be obtained by first assembling the fragments. The assembly process is not perfect, especially in metagenomics, and it very frequently leads to fragments falsely

assembled together (e.g., combining a virus and bacteria fragment together). These false contigs were shown to be detrimental to PhyloPythia's results [21]. Thus its capability to annotate unassembled fragments allows Anacle to avoid this issue.

By allowing fragments to be assigned to a taxon first, we can split the overall assembly task into smaller tasks. Rather than trying to assemble all the fragments at once, we can assemble the fragments in each individual taxon instead. This approach could perhaps reduce the number of false assemblies, as for example the virus and bacteria fragments would end up in different bins and therefore cannot be combined. This then reverses the current method of metagenomic analysis where we first assemble the fragments and then try to annotate the contigs and the remaining unassembled fragments. With Anacle we can annotate the fragments first and then assemble them. The annotation can then be further refined at the end by annotating the resulting contigs of the assembly.

Chapter 7: Conclusion

7.1 Summary of Contributions

We developed an automated system called Anacle to annotate taxonomically the unassembled fragments of a metagnomics project before the assembly process. Knowledge of what proteins can be found in each taxon is built into Anacle by clustering all known proteins of that taxon. The resulting protein clusters can each be represented by profile HMMs. Thus a "skeleton" of the taxon is generated with the profile HMMs providing a summary of the taxon's genetic content. The experiments show that for short, unassembled fragments (100-700 bp), (1) MCL is superior to SOMs in clustering and in running time performance, (2) Anacle achieves good performance in taxonomic annotation, and (3) Anacle has the ability to generalize since it can correctly annotate fragments from genomes not present in the training dataset. Preliminary results on a subset of the unassembled, 100 bp virus fragments from the Sargasso Sea show a dramatic increase in annotation compared to BLAST. Using the typical threshold e-value of 0.001, BLAST only produces hits to 6% of the fragments. Whereas Anacle annotates 63-70% of the fragments, depending on the threshold score. Using our single-threshold results as a reference, this annotation range corresponds to roughly to a TPR of 0.98-0.99 and FPR of 0.02-0.10.

Therefore Anacle is capable of giving quality annotation to short, unassembled fragments, unlike other methods that require the fragments to be assembled first. By allowing fragments to be assigned to a taxon first, we can split the overall assembly task into smaller subtasks. This then reverses steps of the current method of metagenomic analysis.

With Anacle we can annotate the fragments first and then assemble them. The annotation can then be further refined at the end by annotating the resulting contigs of the assembly.

7.2 Future Work

There are many ways in which this work may be extended and applied. In this thesis we focused on the viral genomes, but the same principles can be applied to non-virus genomes. Further work in tuning threshold scores for use with real-world data for both the single and multiple threshold strategies needs to be done. In this thesis the thresholds were tuned to genomes in the classifier's training set. This work can also be applied to a new assembly method where the fragments are annotated first and the resulting taxa are assembled individually, perhaps reducing the number of false assemblies. Finally, the work can be applied to annotating real metagenomics datasets such as the fragments from the Sargasso Sea.

Appendix A: Fragment Dataset Genome Lists

A.1 Genomes used in clustering comparison

The following is the list of genomes used the clustering comparison discussed in Section 5.3. The number scheme here is used in the Figures 14 and 15 of Section 5.3. Genomes #1-10 are herpesviruses and were used for training. All other genomes are from a variety of other viral families.

Genome #	Accession	Description
01	NC_001806.1	Human herpesvirus 1, complete genome
02	NC_001798.1	Human herpesvirus 2, complete genome
03	NC_001348.1	Human herpesvirus 3, complete genome
04	NC_007605.1	Human herpesvirus 4, complete genome
05	NC_006273.1	Human herpesvirus 5 (wild type strain Merlin), complete genome
06	NC_001664.1	Human herpesvirus 6A, complete genome
07	NC_001716.2	Human herpesvirus 7, complete genome
08	NC_003409.1	Human herpesvirus 8, genome
09	NC_007016.1	Macaca fuscata rhadinovirus, complete genome
10	NC_001847.1	Bovine herpesvirus 1, complete genome
11	NC_001653.2	Hepatitis D virus, complete genome
12	NC_007611.1	Baboon polyomavirus 1, complete genome
13	NC_002685.2	Bovine adenovirus D, complete genome
14	NC_003714.1	Pseudomonas phage phi-6 segment S, complete sequence
15	NC_008018.1	Banana streak virus, complete genome
16	NC_004706.1	Papaya leaf curl virus-associated DNA beta, complete genome
17	NC_005031.1	Tomato leaf curl Java virus, complete genome
18	NC_005218.1	Hantaan virus segment S, complete sequence
19	NC_003790.1	Chicken astrovirus, complete genome
20	NC_001782.1	Saharomyces cerevisiae killer virus M1, complete genome
21	NC_007058.1	Bacteriophage ROSA, complete genome

A.2 Genomes used for family cross-validation

The following subsections contain lists of genomes used in the cross-validation tests in Section 5.4.

A.2.1 Herpesviridae

The following lists the genomes used in the cross-validation of *Herpesviridae*. Partition 1, 2 and 3 consists of Genome #1-14, 15-29, and 30-44, respectively. This numbering scheme is used in Figures 16, 19, and 20.

Genome #	Accession	Description
01	NC_006560	Cercopithecine herpesvirus 2, complete genome.
02	NC_001844	Equid herpesvirus 4, complete genome.
03	NC_002577	Gallid herpesvirus 3, complete genome.
04	NC_001347	Human herpesvirus 5 (laboratory strain AD169), complete genome.
05	NC_002665	Bovine herpesvirus 4, complete genome.
06	NC_002686	Cercopithecine herpesvirus 7, complete genome.
07	NC_001826	Murid herpesvirus 4, complete genome.
08	NC_003409	Human herpesvirus 8, genome.
09	NC_002512	Murid herpesvirus 2, complete genome.
10	NC_006150	Cercopithecine herpesvirus 8, complete genome.
11	NC_000898	Human herpesvirus 6B, complete genome.
12	NC_001348	Human herpesvirus 3 (strain Dumas), complete genome.
13	NC_005261	Bovine herpesvirus 5, complete genome.
14	NC_008210	Ranid herpesvirus 2, complete genome.
15	NC_001806	Human herpesvirus 1, complete genome.
16	NC_001798	Human herpesvirus 2, complete genome.
17	NC_001350	Saimiriine herpesvirus 2, complete genome.
18	NC_001847	Bovine herpesvirus 1, complete genome.
19	NC_005264	Psittacid herpesvirus 1, complete genome.
20	NC_007605	Human herpesvirus 4, complete genome.
21	NC_008211	Ranid herpesvirus 1, complete genome.
22	NC_001650	Equid herpesvirus 2, complete genome.
23	NC_002531	Alcelaphine herpesvirus 1, complete genome.
24	NC_006146	Cercopithecine herpesvirus 15, complete genome.
25	NC_006623	Gallid herpesvirus 1, complete genome.
26	NC_001493	Ictalurid herpesvirus 1, complete genome.
27	NC_005881	Ostreid herpesvirus 1, complete genome.
28	NC_006151	Suid herpesvirus 1, complete genome.
29	NC_007646	Ovine herpesvirus 2, complete genome.
30	NC_004065	Murid herpesvirus 1, complete genome.
31	NC_004812	Cercopithecine herpesvirus 1, complete genome.
32	NC_006273	Human herpesvirus 5 (wild type strain Merlin), complete genome.
33	NC_001491	Equid herpesvirus 1, complete genome.
34	NC_002229	Gallid herpesvirus 2, complete genome.
35	NC_007016	Macaca fuscata rhadinovirus, complete genome.
36	NC_003521	Chimpanzee cytomegalovirus, complete genome.
37	NC_003401	Cercopithecine herpesvirus 17, genome.
38	NC_001664	Human herpesvirus 6, complete genome.
39	NC_004367	Callitrichine herpesvirus 3, complete genome.
40	NC_002641	Meleagrid herpesvirus 1, complete genome.
41	NC_001987	Ateline herpesvirus 3, complete genome.
42	NC_007653	Cercopithecine herpesvirus 16, complete genome.
43	NC_002794	Tupaia herpesvirus, complete genome.
44	NC_001716	Human herpesvirus 7, complete genome.

A.2.2 Bromoviridae

This list shows the genomes used in the *Bromoviridae* cross-validation test. Partition 1, 2 and 3 consists of Genome #1-7, 8-15, and 16-23, respectively. This numbering scheme is used in Figure 17.

Genome #	Accession	Description
01	NC_001495	Alfalfa mosaic virus RNA 1, complete sequence.
02	NC_003838	Tomato aspermy virus RNA 2, complete sequence.
03	NC_004120 S	pring beauty latent virus RNA 1, complete sequence.
04	NC_003546	Citrus leaf rugose virus RNA 3, complete sequence.
05	NC_008037	Prune dwarf virus RNA 2, complete sequence.
06	NC_003568	Elm mottle virus RNA 2, complete sequence.
07	NC_003671	Olive latent virus 2 RNA 3, complete sequence.
08	NC_003541	Cowpea chlorotic mottle virus RNA 2, complete sequence.
09	NC_003464	Apple mosaic virus RNA 1, complete sequence.
10	NC_002026	Brome mosaic virus RNA 1, complete sequence.
11	NC_003842	Tobacco streak virus RNA 2, complete sequence.
12	NC_003808	Spinach latent virus RNA 1, complete sequence.
13	NC_004362	Prunus necrotic ringspot virus RNA1, complete sequence.
14	NC_005848	Parietaria mottle virus RNA 1, complete sequence.
15	NC_006566	Fragaria chiloensis latent virus RNA 1, complete sequence.
16	NC_006999	Cassia yellow blotch virus RNA1, complete sequence.
17	NC_003833	Tulare apple mosaic virus RNA1, complete sequence.
18	NC_002038	Peanut stunt virus RNA 1, complete sequence.
19	NC_004006	Broad bean mottle virus RNA 3, complete sequence.
20	NC_001440	Cucumber mosaic virus RNA 3, complete sequence.
21	NC_003451	American plum line pattern virus RNA1, complete sequence.
22	NC_006064	Humulus japonicus latent virus, complete genome.
23	NC_003649	Pelargonium zonate spot virus RNA 1, complete sequence.

A.2.3 Poxviridae

This list shows the genomes used in the *Poxviridae* cross-validation test. Partition 1, 2 and 3 consists of Genome #1-7, 8-14, and 15-22, respectively. This numbering scheme is used in Figure 18.

Genome #	Accession	Description
01	NC_003027	Lumpy skin disease virus NI-2490, complete genome.
02	NC_001993	Melanoplus sanguinipes entomopoxvirus, complete genome.
03	NC_005337	Bovine papular stomatitis virus, complete genome.
04	NC_001611	Variola virus, complete genome.
05	NC_001731	Molluscum contagiosum virus, complete genome.

06	NC_005336	Orf virus, complete genome.
07	NC_003663	Cowpox virus, complete genome.
08	NC_002642	Yaba-like disease virus, complete genome.
09	NC_006998	Vaccinia virus, complete genome.
10	NC_004002	Sheeppox virus 17077-99, complete genome.
11	NC_005309	Canarypox virus, complete genome.
12	NC_006966	Mule deer poxvirus, complete genome.
13	NC_003310	Monkeypox virus, complete genome.
14	NC_004105	Ectromelia virus, complete genome.
15	NC_003389	Swinepox virus, complete genome.
16	NC_001266	Rabbit fibroma virus, complete genome.
17	NC_001132	Myxoma virus, complete genome.
18	NC_002520	Amsacta moorei entomopoxvirus, complete genome.
19	NC_005179	Yaba monkey tumor virus, complete genome.
20	NC_008291	Taterapox virus, complete genome.
21	NC_003391	Camelpox virus, complete genome.
22	NC_002188	Fowlpox virus, complete genome.

A.3 Genomes used in multifamily test

The following lists the genomes in the fragment dataset used in the experiments in Section 5.5. The numbering scheme corresponds with Figures 21-28. This is an extension of the genomes listed in Appendix A.1. Genomes #1-12 are herpesviruses. Genomes #13-24 and #25-36 are bromoviruses and poxviruses, respectively. Genomes #37-47 are from of an assortment of viruses from other families.

Genome #	Accession	Description
01	NC_001806.1	Human herpesvirus 1, complete genome
02	NC_001798.1	Human herpesvirus 2, complete genome
03	NC_001348.1	Human herpesvirus 3, complete genome
04	NC_007605.1	Human herpesvirus 4, complete genome
05	NC_006273.1	Human herpesvirus 5 (wild type strain Merlin), complete genome
06	NC_001664.1	Human herpesvirus 6A, complete genome
07	NC_001716.2	Human herpesvirus 7, complete genome
08	NC_003409.1	Human herpesvirus 8, genome
09	NC_007016.1	Macaca fuscata rhadinovirus, complete genome
10	NC_001847.1	Bovine herpesvirus 1, complete genome
11	NC_001493.1	Ictalurid herpesvirus 1, complete genome
12	NC_008210.1	Ranid herpesvirus 2, complete genome
13	NC_004362	Prunus necrotic ringspot virus RNA1, complete sequence.
14	NC_003546	Citrus leaf rugose virus RNA 3, complete sequence.
15	NC_003842	Tobao streak virus RNA 2, complete sequence.
16	NC_002026	Brome mosaic virus RNA 1, complete sequence.
17	NC_005848	Parietaria mottle virus RNA 1, complete sequence.
18	NC_004006	Broad bean mottle virus RNA 3, complete sequence.
19	NC_003451	American plum line pattern virus RNA1, complete sequence.
20	NC_006566	Fragaria chiloensis latent virus RNA 1, complete sequence.

21	NC_002038	Peanut stunt virus RNA 1, complete sequence.
22	NC_003833	Tulare apple mosaic virus RNA1, complete sequence.
23	NC_003671	Olive latent virus 2 RNA 3, complete sequence.
24	NC_001440	Cucumber mosaic virus RNA 3, complete sequence.
25	NC_008291	Taterapox virus, complete genome.
26	NC_004002	Sheeppox virus 17077-99, complete genome.
27	NC_002520	Amsacta moorei entomopoxvirus, complete genome.
28	NC_001266	Rabbit fibroma virus, complete genome.
29	NC_002642	Yaba-like disease virus, complete genome.
30	NC_001132	Myxoma virus, complete genome.
31	NC_001611	Variola virus, complete genome.
32	NC_001731	Molluscum contagiosum virus, complete genome.
33	NC_002188	Fowlpox virus, complete genome.
34	NC_003391	Camelpox virus, complete genome.
35	NC_003310	Monkeypox virus, complete genome.
36	NC_003663	Cowpox virus, complete genome.
37	NC_001653.2	Hepatitis D virus, complete genome
38	NC_007611.1	Baboon polyomavirus 1, complete genome
39	NC_002685.2	Bovine adenovirus D, complete genome
40	NC_003714.1	Pseudomonas phage phi-6 segment S, complete sequence
41	NC_008018.1	Banana streak virus, complete genome
42	NC_004706.1	Papaya leaf curl virus-associated DNA beta, complete genome
43	NC_005031.1	Tomato leaf curl Java virus, complete genome
44	NC_005218.1	Hantaan virus segment S, complete sequence
45	NC_003790.1	Chicken astrovirus, complete genome
46	NC_001782.1	Saharomyces cerevisiae killer virus M1, complete genome
47	NC_007058.1	Bacteriophage ROSA, complete genome

References

[1] J. A. Fuhrman and L. Campbell, "Microbial microdiversity," *Nature*, vol. 393, pp. 410-411, 1998.

[2] M. Breitbart, P. Salamon, and e. al., "Genome analysis of uncultured marine viral communities," *Proc. Natl. Acad. Sci. USA*, vol. 99, pp. 14250-5, 2002.

[3] M. Breitbart and F. Rohwar, "Here a virus, there a virus, everywhere the same virus?," *TRENDS in Microbiology*, vol. 13, pp. 278-284, 2005.

[4] S. F. Altschul, W. Gish, and e. al., "Basic local alignment search tool," *J. Mol. Bio.*, vol. 215, pp. 403-410, 1990.

[5] R. Dahm, "Friedrich Miescher and the discovery of DNA," *Dev Biol*, vol. 278, pp. 274-88, 2005.

[6] P. Levene, "The structure of yeast nucleic acid," *J Biol Chem*, vol. 40, pp. 415-24, 1919.

[7] A. Hershey and M. Chase, "Independent functions of viral proteins and nucleic acid in growth of bacteriophage," *J Gen Physiol*, vol. 36, pp. 39-56, 1952.

[8] J. Watson and F. Crick, "Molecular structure of nucleic acids: a structure for deoxyribose nucleic acid," *Nature*, vol. 171, pp. 737-8, 1953.

[9] N. A. Campbell and J. B. Reece, *Biology*, Sixth ed. San Francisco: Benjamin Cummings, 2002.

[10]F. Crick, "Central dogma of molecular biology," *Nature*, vol. 227, pp. 561-563, 1979.

[11]C. Woese, O. Kandler, and M. Wheelis, "Towards a natural system of organisms: proposal for the domains Archaea, Bateria, and Eucarya," *Proc. Natl. Acad. Sci. USA*, vol. 87, pp. 4576-4579, 1990.

[12]N. R. Pace, D. A. Stahl, and e. al., "Analyzing natural microbial populations by rRNA sequences," *ASM News*, vol. 51, pp. 4-12, 1985.

[13]K. Chen and L. Patcher, "Bioinformatics for whole-genome shotgun sequencing of microbial communities," *PLoS Comp. Bio.*, vol. 1, pp. 106-112, 2005.

[14]J. C. Venter, K. Remington, and e. al., "Environmental genome shotgun sequencing of the Sargasso Sea," *Science*, vol. 304, pp. 66-74, 2004.

[15]M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt, "A model of evolutionary change in proteins," in *Atlas of Proteins Sequence and Structure*, vol. 5, M. O. Dayhoff, Ed. Washington, DC: National Biomedical Research Foundation, 1978.

[16]S. Henrikoff and J. G. Henrikoff, "Amino acid substitution matrices from protein blocks," *Proc. Natl. Acad. Sci. USA*, vol. 89, pp. 10915-10919, 1992.

[17]S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *J. Mol. Bio.*, vol. 48, pp. 443-453, 1970.

[18]T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J. Mol. Bio.*, vol. 147, pp. 195-197, 1981.

[19]S. Karlin and S. F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes," *Proc. Natl. Acad. Sci. USA*, vol. 87, pp. 2264-2268, 1990.

[20]Y. A. Goo, J. Roach, and e. al., "Low-pass sequencing for microbial comparative genomics," *BMC Genomics*, vol. 5, pp. 3, 2004.

[21]A. C. McHardy, H. G. Martin, A. Tsirigos, P. Hugenholtz, and I. Rigoutsos, "Accuruate phylogenetic classification of variable-length DNA fragments," *Nature Methods*, vol. 4, pp. 63-72, 2007.

[22]T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2001.

[23]H.-H. Bock, "Probability models and hypotheses testing in partitioning cluster analysis," in *Clustering and Classification*, P. Arabie, L. J. Hubert, and G. D. Soete, Eds. River Edge, NJ: World Scientific, 1996.

[24]R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a dataset via the gap statistic," *Journal of the Royal Statistical Society: Series B*, vol. 63, pp. 411-423, 2001.

[25]E. A. Ferran, B. Pflugfelder, and P. Ferrara, "Self-organized neural maps of human proein sequences," *Protein Sciece*, vol. 3, pp. 507-521, 1994.

[26]T. Kohonen, "Self-organizing formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59-69, 1982.

[27]S. Haykin, *Neural networks: a comprehensive foundation*. NJ: Prentice Hall, 1999.

[28]A. J. Enright, S. V. Dongen, and C. A. Ouzounis, "An efficient algorithm for large-scale detection of protein families," *Nucl. Acids Res.*, vol. 30, pp. 1575-1584, 2002.

[29]J. Falkner, F. Rendi, and H. Wolkowicz, "A computational study of graph partitioning," *Mathematical Programming*, vol. 66, pp. 211-239, 1994.

[30]A. Krogh, M. Brown, and e. al., "Hidden Markov Models in computational biology: applications to protein modeling," *J. Mol. Bio.*, vol. 235, pp. 1501-1531, 1994.

[31]S. R. Eddy, "Profile hidden Markov models," *Bioinformatics*, vol. 14, pp. 755-763, 1998.

[32]M. Gribskov, A. D. McLachlan, and e. al., "Profile analysis: detection of distantly related proteins," *Proc. Natl. Acad. Sci. USA*, vol. 84, pp. 4355-4358, 1987.