Efficient Algorithms for Large Data Sets of Genomic Sequences in Microbial Community Analysis

by
David A Knox
B.A., Distributed Studies, University of Colorado at Boulder, 1982

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Masters of Science in Computer Science
Department of Computer Science
2010

This thesis entitled:

Efficient Algorithms for Large Data Sets of Genomic Sequences in Microbial Community Analysis

written by David A Knox
has been approved for the Department of Computer Science

_____
Dr. Robin Dowell

_____
Dr. Debra Goldberg

_____
Dr. Douglas Sicker

Date_____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Knox, David A. (M.S., Computer Science)

Efficient Algorithms for Large Data Sets of Genomic Sequences in Microbial Community Analysis

Thesis directed by Dr. Robin Dowell

Microbial analysis of environmental samples uses high-throughput genomic sequencing to determine the diversity and quantity of microbial species. Current sequencing techniques can produce very large data sets that are not handled by current analysis applications, necessitating the design of better approaches. This work presents three new applications: SeqCluster, ParsInsert, and PTreeView. SeqCluster groups sequences based on similarity using a hierarchical clustering method and selects a representative sequence to create operational taxonomic units (OTUs). SeqCluster also supports large distance matrixes exceeding the size of available local memory by using a custom memory management system. ParsInsert introduces an algorithm that can exploit the knowledge provided by publicly available curated phylogenetic trees to efficiently produce both a phylogenetic tree and taxonomies for unknown sequences. PTreeView is a user-friendly visualization application with a broad range of functions and capabilities supporting very large trees. The applications presented here handle hundreds of thousands of sequences efficiently for data clustering, phylogenetic tree building, taxonomic classification, and tree visualization.

Dedication

To my wife Robyn

There are many people who have moved through my life, but only one who has shared the journey with me. Without your support, companionship, editing skills, and encouragement to reach for my often peculiar goals, I would never have achieved so many of my life's aspirations.

Acknowledgements

There are many people who have supported and guided me to completion of this thesis. I would like to thank Bruce Holland, Incubix Inc, Boulder, Colorado, for allowing me time to explore the fascinating world of Biology, and Charles Robertson, Pace Lab, University of Colorado, for defining the requirements and giving me valuable feedback on the implementation of the clustering and visualization applications.

I would also like to thank my classmates, Antonio González Peña, Kelly Ramirez, Chris Funk, and Alex Poole, who have spent their time in discussions, on projects, and giving feedback for presentations.

I would like to thank my thesis committee, Dr. Goldberg for introducing me to Computational Biology, Dr. Sicker for his insight and guidance in my pursuit of advanced degrees, and my chair Dr. Dowell for her patience and guidance in completing this work.

Finally, I would like to thank all the professors from whom I have taken classes at the University of Colorado, Boulder, in the last 33 years.

Contents

Tables

Figures

1. Introduction

There are bacterial communities all around us in the soil, oceans, waterways, and even within the human body which are vital to everyday human life. These microbiomes contain diverse groups of bacteria working in unison to maintain healthy environments. When the environments are altered by pollution or disease, the community changes in both dramatic and subtle ways. Determining the makeup of the community and identifying significant changes between communities has only recently been possible. High-throughput DNA sequencing technology has supplied much more information than can easily be interpreted by a researcher without computational support.

Determining the number of different species and the relative quantities of each species cannot be performed by counting each unique sequence because sequences from individuals of the same species differ by as much as 3%. The sequence data must be clustered to obtain groups of similar sequences, which represent unique species. Researchers are also interested in knowing the taxonomic classification of each species to help identify function within the community. They want to ascertain the evolutionary distances between all the species in the community in order to understand both the range of diversity and relative differences in microbiome diversity. Phylogenetic analysis produces a tree representing the common ancestry and evolutionary distances between members of the community. The number of species within a microbiome may exceed 10,000, making visualization of the trees difficult without visualization applications.

This thesis explores innovative solutions for solving computational needs for clustering data, taxonomic classification of sequences, phylogenetic analysis, and the visualization of the large data sets produced for studying bacterial communities.

SeqCluster groups sequences based on similarity using a hierarchical clustering method and selects a representative sequence from each group to create operational taxonomic units (OTUs). Using a custom memory management method, SeqCluster supports large distance matrixes that exceed the size of available local memory.

ParsInsert introduces an algorithm that can exploit the knowledge provided by publicly available curated phylogenetic trees to efficiently produce both a phylogenetic tree and taxonomies for unknown sequences.

PTreeView is a user-friendly application to visualize and customize phylogenetic trees. It has a broad range of functions and capabilities, such as formatting, zooming, searching capabilities, clade support with topology manipulation, bootstrap support, custom annotations, and support for large trees containing hundreds of thousands of taxa.

These innovative and efficient solutions handle the large data sets produced by current high-throughput sequencing technology by addressing the computational needs for clustering data, taxonomic classification of sequences, phylogenetic analysis, and tree visualization.

2. Background

Complex microbial environments are being studied to discover the differences in composition and abundance of microbes in healthy and unhealthy communities. Understanding the full breadth and depth of this microbial diversity provides valuable insight into the biology of global processes. To study these microbiomes, the diversity and abundance of each microbe type must be determined within a sample. There are millions of bacteria in each sample from many species, yet it has been estimated that microbiologists have been able to culture in a laboratory less than 1% of all bacteria [1].

Determining the number of species and relative quantities of each species was, until recently, a manual task that required counting the number of each bacteria type through microscopic images or depositing each bacterium onto a growth media and allowing it to culture. Many of the bacteria in a microbiome have such low quantities that they were never selected, while others were selected but did not clone themselves in the growth media. These factors kept past researchers from determining the "true" diversity of the microbiome.

2.1. Sampling

Biologists investigating natural bacterial communities are confronted with the problem of knowing how well the sampling represents the true diversity of the microbiome. It has been estimated that there are billions of bacteria in a handful of soil [1]. A variety of statistical methods have been used to determine how much of the community has been sampled by looking at the number of species and the occurrence of those species [2, 3].

The advent of current sequencing techniques has made technology available to easily analyze whole communities and the cost per sequence continues to fall. The Polymerase Chain Reaction (PCR) method allows DNA from any organism to be amplified and allows detection of DNA from a single organism in the sample. Current techniques of Sanger, 454, and Illumina sequencing allow samples to be analyzed in a matter of hours and can produce hundreds of thousands of unique sequences.

DNA is made up of four different nucleotides in a double stranded helix structure. The nucleotides adenosine (A), guanine (G), cytosine (C), and thymine (T) form canonical Watson-Crick base pairing across the helix (A-T, C-G). Heat causes the double stranded helix to separate into single strands, and when free nucleotides and polymerase are available as the sample is cooling, the polymerase will add complementary nucleotides to the single strands, creating two identical double helix strands. PCR doubles the number of copies during each thermal cycle, allowing the detection of even low abundance members after only a few iterations.

Current sequencing methods capture single stranded DNA, replicate the strand by PCR, and use special nucleotides that will fluoresce when incorporated into the complementary sequence. Each of the nucleotide types is added one at a time, and if the next nucleotide required to build the complementary sequence is being added, the sample will fluoresce. The process is repeated along the whole length of the single strand and records which nucleotide type fluoresces to obtain the "read" for that sequence.

Current sequencing techniques can produce "reads" of the sequences simultaneously for up to one million different fragments of DNA per processing run. How

well this procedure has captured the overall variety of the sample can only be estimated by determining the quantity and distribution of distinct types of organisms found in the sample. Determining the sequence taxonomy and quantities of each species allows the estimation of the number of unseen species and determines the coverage of the sample [2]. Distributions of different samples can be compared to distinguish significant changes between samples.

2.2. Phylogenetic Analysis

Phylogenetics is the study of evolutionary relationships between groups of organisms. The DNA for a species is a dynamic system, changing to keep the species viable in changing environments. As different groups of organisms change, they can diverge and create new species. Phylogenetic analysis attempts to discover the underlying relationship between species by finding the most likely common ancestors of a set of sequences. These relationships are often shown graphically in phylogenetic trees, such as the Tree of Life. The Tree of Life, Figure 1, shows the overall relationship of all branches of life on this planet. Phylogenetic analysis is used to determine the relationship between all sequences within a sample, allowing biologists to view the overall composition and diversity of organisms in the environment under study.

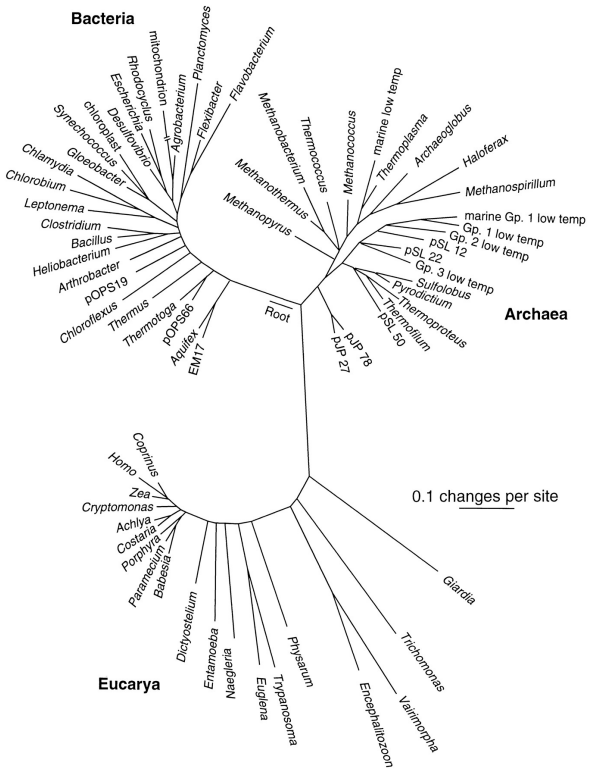

Figure 1 - Tree of Life showing evolutionary divergence of species. Branch points represent common ancestors of the descendants. Length of the branches represents evolutionary time. [1]

PCR can be used to selectively amplify the sequences from a single gene through careful selection of short synthetic DNA fragments called primers. BY selecting primers that can only match DNA from a known gene, only the DNA from that gene is replicated for sequencing.

Phylogenetic analysis of a community requires that the gene selected for sequencing must exist in all living organisms, be conserved enough to show relationships between evolutionary distant relatives, and variable enough to differentiate more recent evolution. The most common gene sequenced is the Small Ribosomal Subunit, SSU rRNA, since it is required for protein production in all microbial organisms. The SSU rRNA gene is a short sequence of approximately 1500 nucleotides and has both variable and highly

conserved regions. Phylogenetic analysis can be used to distinguish sequences down to the species level. Since the ends of the sequence are conserved regions, PCR can be performed using universal primers that do not require knowledge about the community before processing.

As valuable as SSU rRNA has been for phylogenetic analysis, it does have its limitations. Because it is vital to every organism's survival, there are many copies of the gene sequence in every genome. The number of copies varies and can cause over/under representation in sequencing data. Despite the limitations, using SSU rRNA for analysis is a powerful method and has discovered many organisms that were previously unseen.

2.3. Operational Taxonomic Units

Many applications require the species counts and abundance counts to determine if significant differences exist between samples. The data is clustered to create Operational Taxonomic Units (OTU) because the amount of data being collected by current sequencing methods can often overwhelm the analysis algorithms. The counts of sequences in each OTU, along with a single representative sequence from each OTU, are used for further analysis. Clustering algorithms are used to combine the similar sequences until the diversity reaches a threshold at which members of a common species diverge. The threshold is commonly set at 97% sequence similarity for the SSU rRNA sequence[4].

2.4. Taxonomy

Taxonomy is the classification, identification, and naming of organisms. Classification was first performed by looking at the morphological features of species and grouping them by commonality of those features. For example, wings vs. four legs, fur vs. scales, or two legs vs. four legs. With current sequencing methods, the visible genotypes

are not required since the sequences of related species provides better evidence than the physical traits[1].

Taxonomy is used in biology to combine similar individual sequences into groups. The taxonomic classifications are a hierarchical set of categories referred to as ranks. In the ordered set of ranks (Kingdom, Phylum, Class, Order, Family, Genus, and Species), Kingdom is the most general category and Species is the most specific category. Within any species, individuals will have slightly different sequences for a given gene. The amount of difference allowed is dependent on the gene being sequenced, but is often set at 3% for SSU rRNA[4].

Assigning taxonomic classification can be obtained from BLAST, a database searching algorithm provided by the National Center for Biotechnology Information (NCBI). The database contains all the known taxa (species or subspecies) sequences. Each taxon has a database record annotating the known information. BLAST finds the most homologous sequence matches in the database and assigns the taxonomy based on the best matching sequence. Sometimes the most significant match returned by BLAST is not the best taxonomical match and can lead to erroneous taxonomy assignments. Machine learning methods, such as Hidden Markov Models (HMM) and Support Vector Machines (SVM), are fast and excellent classifiers for the sets of taxonomies on which they are trained. However, they have difficulty with sequences that exhibit similar sequence patterns with species from different branches of the taxonomy. I have implemented a method for taxonomic classification that infers the taxonomy for an unknown sequence by inserting it into a phylogenetic tree created from sequences with known taxonomy.

2.5. Phylogenetic Trees

Although many biologists desire phylogenetic trees for further analysis, the tree building methods have exponential runtime complexity. Building trees can take hours for a few hundred taxa, which makes processing 100,000 sequences impractical. There are many databases that are dedicated to SSU rRNA and some provide curated phylogenetic trees. Curation adds knowledge to the tree that cannot be captured by the sequences alone. Greengenes [5] provides a "core" tree of approximately 10,000 taxa that are considered a good representation of the diversity of known taxa. This work introduces an algorithm that can exploit the knowledge provided by these curated trees and produces both a phylogenetic tree and taxonomies for unknown sequences.

Parsimony is a phylogenetic analysis based on the assumption that the minimal amount of change between sequences represents the path of the last common ancestor or last divergence point[6]. Parsimony techniques may be more sufficient at classifying unknown sequences [7] than other methods, such as HMM[8], SVM[9], or Bayesian[10] techniques. One of the goals of this work was to design a program to parsimoniously insert a large number (>100,000) of environmental sequences into a core phylogenetic tree. While parsimony may provide one of the best methods of phylogenetic analysis, few programs can handle multiple thousands of sequences. Current sequencing technology will only increase the number of sequences produced by each sequencing run, requiring that better tools be designed to handle this large amount of data.

Phylogenetic tree building algorithms are usually very time consuming when handling large numbers of sequences. To make the algorithms more usable, the amount of data needs to be reduced. There are many sequences that are nearly identical and the most

common method for data reduction is to create OTU's by clustering the sequences. There are a number of different applications that can be used for clustering[5, 11, 12]. If the taxonomy of the sequences is known, clusters can be selected by sorting sequences with the same taxonomy into groups. Other methods, such as agglomerative hierarchical clustering, can be used to group by sequence similarity.

2.6. Clustering

2.6.1. Clustering Methods

There are two major types of clustering methods: hierarchical and partitional. Partitional clustering is a classification problem that assigns a set of observations into clusters with other similar observations. The number of clusters is usually predetermined and the method finds the best observations to place in the number of given clusters. Hierarchical clustering is a stepwise method that makes a decision at each step for creating a cluster. The final output from this method will be a tree showing the relationship of all the observations. Hierarchical clustering can either be agglomerative or divisional. Divisional places all observations into a single cluster and finds the best way to divide the set into two. It then iterates on each cluster until each contains a single observation, making it a top-down method. Agglomerative takes the alternate bottom-up approach where each observation is placed in its own cluster. At each step the two most similar clusters are merged into a single cluster. The order in which clusters are merged determines the tree structure, where the root of the tree represents a single cluster encompassing all observations. The agglomerative hierarchical method was used in the implementation of algorithms for this thesis.

2.6.2. Distance Matrix

Hierarchical clustering algorithms are powerful because any comparison function can be used to determine similarity between any two observations. The similarity is usually referred to as a distance between observations. The distances of all pairs of observations are placed into a distance matrix, which is used to determine the next clustering step.

Once two clusters are selected as the next to be merged into a new cluster, the distances from the new cluster to all other clusters must be determined. There are three common methods: single-linkage, average-linkage and full-linkage. Each method creates clusters with different properties. To find the distance from one cluster to another, single-linkage finds the minimum distance from any member of one cluster to any member of the other cluster. This allows the cluster to follow patterns within patterns, such as concentric circles. This method is the fastest, but tends to drift through the data creating diffused similarity within the cluster. Average-linkage tries to mediate between the two other methods, using an average of the distances from one cluster to the other. Full-linkage creates clusters that are the most tightly bound by using the maximum distance between members of the clusters. It cannot see patterns of overlapping similarity and would fail at finding concentric circles, instead taking a fraction of each circle as part of a cluster. Full-linkage is most useful for phylogenetic analysis because of the tight similarity in the clusters produced.

2.7. Visualization

Visualization of large datasets is difficult with limited available screen space. The information is complex, and exploring the data without the use of graphical interfaces is time consuming and overwhelming. Tree organization lends itself to graphical display. The

common information can be collapsed into a representation of clusters or clades. Hiding and highlighting information is required to allow researchers to focus attention on specific details of a particular tree.

3. Related Work

3.1. Clustering

Most of the previous work in clustering large data sets has been in the data mining research area [13] and recently applied to clustering of sequence data. Creation of Operational Taxonomic Units (OTUs) has seen a revival of clustering methods to reduce the number of sequences used for phylogenetic analysis[11, 14, 15]. Fast clustering algorithms available today cannot support the large datasets that are produced by modern clustering because they take $O(N^3)$ to $O(2^N)$ in time and require $O(N^2)$ in space, which will exceed the address limits of 32 bit machines. For smaller data sets there are a number of integrated application suites that have implemented data reduction methods[16, 17].

The SSU rRNA gene has been used for sequencing to determine diversity of environmental samples. There are multiple databases designed to identify and annotate SSU rRNA gene sequences, such as NCBI, Greengenes, and SILVA, each with its own set of curation tools. Greengenes, for example, provides sequence alignment for "browsing, blasting, probing and downloading" [5]. The main goals for sequence analysis of unknown organisms typically involves identification (taxonomy) and tree placement (phylogenetic relatedness). There are a number of ways to classify and insert unknown sequences into a "core" tree [7]. Greengenes uses a BLAST method that compares NAST aligned sequences against a comprehensive and up-to-date database of bacterial and archaeal 16S rRNA genes with taxonomy determined by multiple curators[18]. SILVA uses a cross-checked dynamic alignment process that gives each sequence an alignment score that is used to identify taxonomy [19]. The Naïve Bayesian classifier developed for the Ribosomal Database Project

(RDP) uses a word-based (8 base pair) algorithm using oligonucleotide frequencies to classify sequences according to Bergy's Manual [20]. ARB directly inserts sequences into a core phylogenetic tree through parsimonious insertion [21].

Parsimony is a phylogenetic analysis based on the assumption that the minimal amount of change between sequences represents the path of the last common ancestor or last divergence point[6]. Parsimony algorithms have been shown to work well on large data sets as they provide better phylogenetic distance results in more accurate diversity measurements and parsimony techniques may be more sufficient at classifying unknown sequences [7]. However, the use of parsimony is currently only used by ARB which provides methods for annotating trees and a tree-viewing program [21]. The insertion of unknown sequences is time consuming, 250 per hour on a 2010 computer. ARB software uses a database of ~30,000 sequences and inserts new sequences into the core tree using ARB-parsimony insertion and branch lengths are superimposed onto a parsimony-generated tree.

3.2. Tree Building

There are many tree building applications available which use a variety of algorithms, from simple distance methods to parsimony, Bayesian[3], and maximum likelihood methods. Parsimony algorithms were introduced in the 1970's by Fitch [22]. One of the early contributors of applications, Felsenstein, maintains a web page, listing not only the large library of phylogenetic applications he has contributed to, but also links to all of the available applications [http://evolution.genetics.washington.edu/phylip.html]. The parsimony concepts for increasing the speed of parsimony methods are described in [23] and for handling larger data sets in [24, 25].

Algorithms, such as maximum likelihood, that search all possible tree configurations produce the most accurate results, but are $O(2^n)$ in time which limits the number of sequences that can be processed in a reasonable amount of time. Branch and bound methods seek to limit the number of possibilities, but still require massive computational resources.

3.3. Visualization

There are many alternative tree viewing applications available. Table 1, adapted from [26], shows commonly used visualization applications. Only Dendroscope, PHYLIP's drawtree, and TreeJuxtaposer can handle large trees. Many are limited to the manipulations that can be performed. This is an active field of research because there is a need for creating trees that can be used in published articles and for dynamically exploring phylogeny.

Table 1: Features of visualization applications.

	Taxa	Search	Compare	Color	Edit	Collapse Subtree	Rerooting
PTreeView (this work)	Limited by memory	✓	✓	✓	✓	✓	✓
Dendroscope[27]	350k	✓		✓	✓	✓	✓
HyperTree[28]	20k	✓		✓			
MEGA[29]	20k			✓		✓	✓
PHYLIP[30]	1336k						
TreeDyn[31]	5k	✓		✓	✓	✓	✓
TreeJuxtaposer[32]	1002k	✓	✓	✓		✓	
TreeView[33]	2k			✓	✓	✓	✓

4. SeqCluster – Operational Taxonomic Unit Clustering

SeqCluster application was created for use in the Pace Lab at the University of Colorado, Boulder, to help with the analysis of microbial data. It was designed to handle large data sets up to 100,000 sequences, perform efficiently, and produce outputs at multiple points during the clustering process to provide different sets of OTUs to be used for further processing. These were accomplished by using a hierarchical clustering method based on similarity of the sequences.

SeqCluster performs hierarchical clustering using single, full, or average linking. The implementation was optimized to handle large data sets, utilizing multiple processors and multiple threads per processor to minimize idle times due to waiting on disk reads/writes.

Modern sequencing can produce hundreds of thousands to millions of reads in a single day. Data analysis requires that this large amount of information be reduced to a more manageable amount by the creation of Operational Taxonomic Units (OTUs). SeqCluster creates OTUs by clustering the similar sequences into a group and selecting a representative from the group to be used for further processing. Clustering is accomplished by means of an agglomerative hierarchical algorithm using a distance matrix comprised of sequence similarity. Clustering is continued until only one group remains.

Sequence similarity is calculated by finding the percent of matching nucleotides between two sequences while ignoring insertions and deletions. Because the sequences are from a very slowly changing gene, near relatives will have a similar alignment and matching set of nucleotides and more distant relatives will have fewer matching nucleotides. The

sequence comparison is usually the time limiting step when clustering a large set of sequences, even though the algorithm is $O(cN^2)$ in time and space. Once the matrix no longer fits in local memory and data must be swapped out to disk, the time constant be comes large, 10^6, and contributes considerably to total execution time. Given a simple 4-byte value to store a distance, 32,000 sequences will require 4 GB to store the distance matrix, which is the full address space of 32 bit computers. To keep a full distance matrix for 100,000 sequences will require over 37 GB. The Greengenes database contains over 300,000 sequences for SSU rRNA and would require 330 GB to store a distance matrix. The first implementation of SeqCluster was developed for Microsoft Windows platform using a 32-bit compiler, which normally limits the data address space to 2GB, but can be extended to 3GB using special initialization code. Consequently, SeqCluster implemented memory management to support data sets which exceeded memory capacity.

4.1. Algorithm

The user can select between single-linkage, full-linkage, and average-linkage methods. The different algorithms provide different cluster characteristics that lead to different members within groups. Single-linkage is the fastest, but tends to wander through the data. Average-linkage produces clusters that are not as tightly contained and is similar in speed to the full-linkage method. Full-linkage is the slowest method, but provides the tightest clusters and is the preferred method for OTU creation.

Simple distance calculations that assume a constant rate of evolution, are known to have long branch attraction, where longer sequences have greater similarity scores because each single nucleotide mismatch is a smaller proportion of the total sequence length. This problem is magnified over larger evolutionary distances when many insertions or deletions

have accumulated. The SSU rRNA sequence is well conserved and is not as affected by this limitation.

The clustering algorithm pseudo code is given in Figure 2. The preprocessing step is performed to create the distance matrix and is saved to disk. The complexity of the preprocessing step is $O(N^2)$ in time and space. The clustering step reads a matrix from disk and creates a copy so the clustering step does not overwrite the original distance data, which is required to calculate the centroids of each group for representative sequence output. This also allows clustering with different methods to be performed without recreating the distance matrix. The clustering step has N-1 steps to completely cluster all sequences. Determining the next row to cluster would take N^2 in a naïve approach, but SeqCluster keeps row header information including the minimum value within the row, therefore only a single pass through the N rows is needed to find the next two groups to cluster. Each clustering step requires updating 2N (one row and one column) distances and updates the minimum value for the row. Overall the time is $O(N^2)$.

```
// Algorithm Pseudo Code
1. Create Distance Matrix
        Read sequences from FASTA input file
        For each sequence
            Add new row to matrix
        Write compressed sequence to random access file, link to row structure
        For each row
            Allocate data for row in a random access file, link to row structure
        If multithreaded
            Create number of threads requested
            Wait while threads not complete
        Else
            For each row
                For each column > current row
                    Calculate the distance between seqeunces
                Write row data to file
        Reflect the upper matrix to create fully populated matrix

2. Clustering
        Copy original distance matrix for cluster processing
        Place all rows into their own group
        For each group, find the minimum distance to another group
        While more than 1 group remains
            Find the minimum distance between any two groups
            If next grouping exceeds output step
                Create tree file
                // Each group is an OTU. Select one representative from each OTU
                For each current group
                    Determine a representative by finding member closest to the groups centroid
                    Write representative sequence to FASTA output file

            Collapse the two rows together
                Distances are updated using user selected method (min, max, avg)
                Remove one of the groups from the list

            Update each groups minimum distance
            Add record to the tree stack

        Write final output file set
```

Figure 2 - Clustering algorithm pseudo code. Creating the distance matrix can be distributed to multiple threads/CPUs. The clustering step repeatedly finds the next two closest clusters and merges them together. The clustering loop completes when there is only one group left. Output of intermediate steps produces sets of OTUs with differing similarity among the members.

4.2. Implementation

4.2.1. Input and Output

SeqCluster reads a set of aligned nucleotide sequences from a FASTA formatted file. Each sequence becomes a row and column in the distance matrix.

SeqCluster creates a set of files at each output point. The OTU list contains a inventory of sequences that are members of each OTU and indicates which sequence is the representative for the cluster. A tree file is created, in Newick format, showing the hierarchy of clustering that has occurred up to that point. A FASTA file is created with all the representative sequences for use in further analysis.

The implementation defaults to creating a set of output files at each percent similarity, at 99%, 98%, 97%, etc. The user chooses the output set that meets their specific needs of either similarity (97% similarity of SSU rRNA is considered the divergence limit of sequences from the same species) or the output set that contains the number of clusters that can be easily processed by other data analysis applications.

4.2.2. Memory Management

Memory management is performed through a simple set of functions for moving data between memory and disk. The application manages the use of the data rows to limit the number of rows in memory at any one time. The application locks and releases the row data as needed. The memory management recovers the buffers from the row data and holds it in a free list. If the same row data is requested, the buffer is removed from the free list.

Data writes are the responsibility of the main application and not the memory management subsystem. The memory management is responsible for abstracting the data location within the disk file and supports partial reads and writes of a single row.

4.2.3. Distance Matrix

Calculation of each position in the distance matrix creation is independent, allowing the processing to be distributed. SeqCluster implemented a method to distribute the processing across multiple threads and CPUs. The hardware available at the time of the implementation had two quad core processors, each with hyper-threading. The implementation used multiple threads per CPU to keep the CPU busy while the other thread was blocked, waiting for disk access. A shared memory segment was used to synchronize memory access to information and the disk IO processor was used to handle overlapping sector writes on the shared matrix file.

4.2.3.1. Sequence Compare

Sequences are represented by an ASCII string using the letters “ACGT” to represent the nucleotides and “-“ to represent gaps in the alignment. Each ASCII character is an 8-bit representation for 3 bits of information. Because memory is a premium, sequences are compressed to minimize local memory storage of the sequence. Using only 3 bits to represent the 5 states, compression can pack multiple nucleotides into a 16-bit word. The compression of the sequences also allows speed enhancements to the comparison function by allowing the processing of multiple sequence positions at the same time. The packed 3-bit values are used as an address into a pre-calculated scoring matrix, shown in Figure 3, requiring only a simple lookup and no calculations per compare. This can be extended to create a larger scoring matrix using 12-bit packed values representing 4 nucleotides.

Although this is only a 4X speed increase, it also has the advantage of keeping the CPU within its instruction cache during execution, which further increases the efficiency of the implementation.

	A	C	G	T	-
A	1 ; 1	0 ; 1	0 ; 1	0 ; 1	0 ; 0
C	0 ; 1	1 ; 1	0 ; 1	0 ; 1	0 ; 0
G	0 ; 1	0 ; 1	1 ; 1	0 ; 1	0 ; 0
T	0 ; 1	0 ; 1	0 ; 1	1 ; 1	0 ; 0
-	0 ; 0	0 ; 0	0 ; 0	0 ; 0	0 ; 0

	AAAA	AAAC	AAAG	AAAT	AAA-	…
AAAA	4 ; 4	3 ; 4	3 ; 4	3 ; 4	3 ; 3	
AAAC	3 ; 4	4 ; 4	3 ; 4	3 ; 4	3 ; 3	
AAAG	3 ; 4	3 ; 4	4 ; 4	3 ; 4	3 ; 3	
AAAT	3 ; 4	3 ; 4	3 ; 4	4 ; 4	3 ; 3	
AAA-	3 ; 3	3 ; 3	3 ; 3	3 ; 3	3 ; 3	
…						

Figure 3 - Scoring matrix for single and multiple positions. The table on the left shows a lookup table when comparing two single characters. The two values in the table are the number of correct matches and the number of comparisons performed. The compare function ignores inserts and deletions, so only positions that both have a nucleotide are compared. Comparing A to a gap '-' does not contribute to either the matches or number of compares. The partial table on the right compares 4 sequence positions at a time. Using the bit packed sequence values to find the position in the lookup table and the comparison function immediately obtains both the number of matches and number of comparisons from the table. Iterating through the sequence four positions at a time, accumulating the values from the table allows a highly efficient calculation of the similarity between two sequences.

To calculate a distance between two sequences, the number of positions that both contain a nucleotide and the number of matching positions are used to calculate the percentage of similarity. The tables in Figure 3 show the pre-calculated values for sequence character comparison. The first value is the number of matches and the second value is the number of positions where there are nucleotides (not gaps) in both strings. Comparing sequences AAAC and AAAT would produce 3 matches in four comparisons, where a comparison of T-GT and TA-T would match two positions in two comparisons. The table on the right compares four positions at a time. The multiple position table using 12-bits per compare requires a 16MB table. The inner loop for comparing two compressed sequences was timed at a mere 9 ns per iteration.

4.2.3.2. Matrix Reflection

The distance calculation from sequence A to B is the same as from B and A, requiring that only half of the matrix be calculated and then reflecting the values onto the other half. Because the comparison function is more time consuming than reading values from disk, a reflecting function for the matrix was implemented. Because the whole matrix may not fit in memory, the matrix is reflected in segments.

4.2.4. Clustering

The basic hierarchical clustering algorithm picks the two groups to cluster, and then reduces the matrix by combining the two rows and updating the columns of the other rows to reflect the distance to the new cluster. When the full matrix will not fit into available memory, disk access to the stored data must be performed. Disk access, which is a million times slower than memory access, greatly increases the time to process. By postponing the update of the columns until needed, there are many updates that can take place together or are completely unneeded as subsequent merges overwrite the data before it is used. This adjustment can eliminate the disk access and increase the execution speed. SeqCluster has implemented this feature and can be optionally enabled.

4.2.5. Benchmarks

SeqCluster was tested using a number of different sized data sets. The execution time and size of the distance matrix are shown in Table 2. The tests were run on a 2.8 GHz Pentium 4 Intel processor running Windows XP in 2 GB of RAM.

Table 2: Time and Space required to process a number of sequences.

Number of sequences	GB distance matrix	Execution Time in Hours
5000	.1	.1 (5 min)
16000	1	.4 (24 min)
32000	4	1.3
96000	34	9.25

4.2.6. Port to Unix

Original development for SeqCluster was on Windows platform, however a port was completed to both 32-bit and 64-bit Unix machines. Some features were eliminated during the port. Memory management was removed because the 64-bit operating system has a large enough address space and more efficient algorithms that are responsible for page swapping. The 8 GB of memory available on the test system allows fast processing of over 40,000 sequences per run.

5. Phylogenetic Tree Building and Taxonomic Classification

Microbial community analysis requires information about the relationship of sequences to each other and taxonomic classification. ParsInsert introduces an algorithm that can exploit the knowledge provided by publicly available curated phylogenetic trees to efficiently produce both a phylogenetic tree and taxonomies for unknown sequences. Test results show that the taxonomic classifications and insertions of sequences into the phylogenetic tree are very accurate.

Once the sequence data of an environmental sample has been analyzed and reduced to a set of species, phylogenetic analysis can be used to understand the evolution and degree of relatedness among species within and between samples [34]. The analysis uses the count of each species in the sample and the degree of relatedness (nearness in a phylogenetic tree) to estimate the significance of differences between microbial communities. The differences between communities containing only slight variations of closely related species are not as significantly different as samples where the species are from more distant relations.

Although phylogenetic trees are desired for further analysis, the tree building methods have exponential runtime complexity. Building trees can take hours for a few hundred taxa, which makes processing 100,000 sequences impractical. As current sequencing methods continue to increase the number of sequences per run, the computational time will become incapacitating. New computational methods are required to support these technology changes.

In microbial analysis, the most commonly used gene for phylogenetic analysis has been SSU rRNA. There are conserved regions to be used for relating distant relatives and variable regions that provide distinction among closer relationships. There are many databases that are dedicated to SSU rRNA and some provide curated phylogenetic trees. Curation adds knowledge to the tree that cannot be captured by the sequences alone. Greengenes [5] provides a "core" tree of approximately 10,000 taxa that are considered a good representation of the diversity of known taxa. This section introduces an algorithm that can exploit the knowledge provided by these curated trees and produce both a phylogenetic tree and taxonomies for unknown sequences in a fast and accurate method.

Curators have evaluated the relationships between the core sequences and have accepted a particular tree as the correct tree. The topology of the tree should not be changed by the sequences being processed, but instead, the sequences should be inserted to pad out the tree. The ParsInsert application was created to take advantage of the knowledge imparted by the curators through the use of parsimonious insertion into a curated tree. ParsInsert only inserts the new sequences into the tree while maintaining the original topology among core sequences. Finding the insertion points for any new sequence is only dependant on the original core tree, thus allowing the insertion process to become distributable. The ParsInsert algorithm also limits the number of comparisons to the size of the core tree, therefore increasing the speed of tree creation. ParsInsert produces a phylogenetic tree with both core and unknown sequences and infers taxonomy for each unknown sequence by examining the insertion location.

Parsimony is a phylogenetic analysis method based on the assumption that the minimal amount of change between sequences represents the path to the last common

ancestor or last divergence point. Parsimony algorithms have been shown to work well on large data sets, as they provide better phylogenetic distance results and more accurate diversity measurements. Parsimony techniques may also be better at classifying unknown sequences [7]. Building a full tree de novo using parsimony is dependant on the processing order of the sequences. Parsimonious Insertion eliminates the stochastic nature and provides a deterministic process for every insertion into the core tree, irrespective of other sequences to be inserted.

Greengenes provides a full tree containing more than 240,000 sequences found in their database. This includes the 10,270 sequences that form their core taxa. Python scripts were created to automate the preprocessing of the tree by trimming all sequences that were not in the core sequence list while maintaining the total branch lengths from sequences to the root. This is the core tree used in all further references.

Greengenes also maintains annotations for each sequence in the database that include taxonomy assignments from multiple curators: RDP, PACE, LUDWIG, HUGENHOLTZ, and NCBI. The assignments from one of the curators, Ribosomal Database Project (RDP), were selected for use in testing the implementation. The RDP assignments were chosen because they were the most widely available in all the sequences selected for the test sets.

5.1. Algorithm – Parsimonious Insertion into Curated Tree

```
Read in the given a core taxa tree topology, sequences for the core taxa, and taxonomy file.

For each branch node in the tree
        Calculate the parsimony sequence based on the children of the node
        Calculate the taxonomy as intersection of taxonomy from all children

        for each group of new sequences that fit into memory
                for each node in tree
                        for each sequence in memory group
                                calculate score between node and sequence
                                Add score,node pair to best matches for sequence
                                Keep top K scores
                for each sequence in memory group
                        write top scoring positions and taxonomy

Add the sequences into the tree at the positions that matched best
        If best match is a taxa, use the parent of taxa as insertion position
        If only one new sequence is to be added at the branch node
                Add as child
                Calculate distance from branch node to sequence
        Else
                Create new branch node as child
                Add group of sequences as child to new branch
                Calculate distance from branch node to each sequence
                Use half of smallest value as distance from best position and new branch
                Subtract half of smallest value from all children of new branch

 Write new tree with taxonomy annotation
 Write new tree with names only
```

Figure 4 - Algorithm for Parsimonious Insertion into a tree without modification to the tree's topology.

5.1.1. Creating a Sequence for the Common Ancestor

Each common ancestor in the core tree generates its parsimonious sequence from the sum of its descendants. The sequence is stored as a series of 4-bit patterns, where each bit represents the occurrence of A, C, G, or T at that position in one of the descendants. Brackets are used to delineate the set of nucleotides that can occur at any position. For example, the sequence AGGT can be written as [A][G][G][T]. This format can represent more than one nucleotide at a position. The sequence [A][AG][G][T] shows two nucleotides are possible at the second position. Figure 5 shows the accumulation of possible nucleotides at each position from all the descendants. The sequences ACTT and ATTT only

differ in the second position, so their common ancestor's composite sequence can be simply written as A[CT]TT.

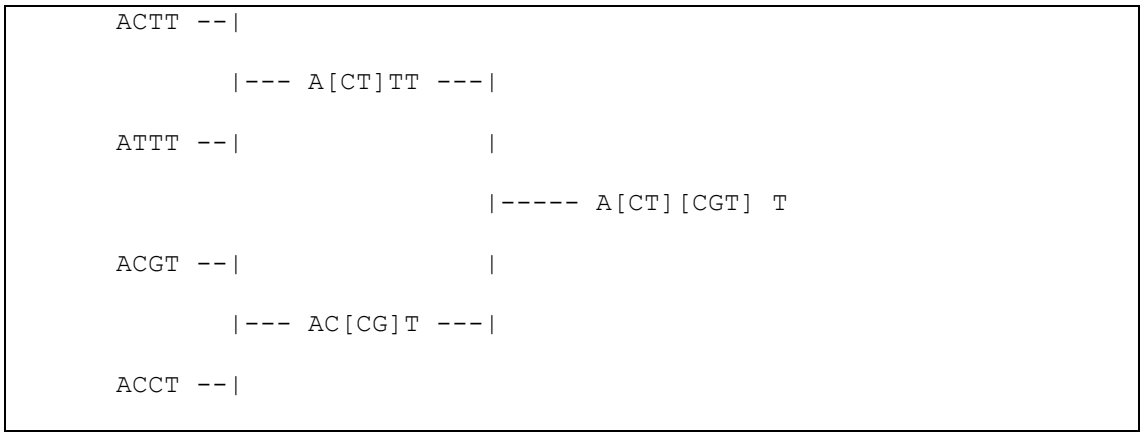

Figure 5 - Example of parsimonious sequences for common ancestors. The composite sequence is an accumulation of all possible nucleotides seen at that position in any of a common ancestor's descendants.

Continuing the process, the next common ancestor sequence results from the combination of A[CT]TT and AC[CG]T to produce A[CT][CGT]T as the composite of all four original sequences. These composite sequences are used to compare internal nodes of the tree and unknown sequences to locate possible insertion points. The best location for insertion into the tree may be at the taxon level or at a common ancestor.

5.1.2. Creating Taxonomic Classification for Common Ancestor

The taxonomy of a common ancestor is inferred from analysis of the taxonomy of all descendants. If there are many different taxonomic values at a given rank, the taxonomy of the common ancestor must not assign that rank or more specific ranks because the diversity implies this common ancestor encompasses multiple groups. ParsInsert uses a simple majority of 50% or better to assign a rank value, which will bias the taxonomy of the tree towards the species that are more highly represented.

5.1.3. Comparison Scoring Function

ParsInsert compares the unknown sequence being inserted to all of the taxa sequences and all the composite sequences at a common ancestor. Comparison of a sequence to a composite sequence must take into account the different possible nucleotides at each composite position, allowing a composite match if the composite contains the nucleotide from the sequence being inserted. The scoring function used in ParsInsert is a cost function that is calculated by adding up the number of differences, insertions, deletions, and one-quarter of the composite positions that matched. The composite positions matched are partially weighted to allow sequences with more exact matches to have a lower cost than one that has more matches to composite positions. The scoring function is a cost function where the best scores are the smaller scores. An exact match of sequence would have no cost in this scoring scheme.

5.2. Results

5.2.1. Test Set Generation

Python scripts were written to generate sixteen data sets by randomly selecting taxa from the 230,000+ taxa sequences not already in the "core" tree of the Greengenes database. Generation of the test sets did not consider the availability of an assigned taxonomy from any selected curator, therefore the test sets can be used to test the algorithm independently of the selected curator. During ParsInsert testing, sequences that did not have taxonomy specified by the selected curator were excluded from accuracy measurements. Testing was performed with sixteen sets, ten sets each with 100 taxa, five sets each with 1000 taxa, and one set with 10,000 taxa.

The test results may show some bias because of the random selection of taxa from a skewed data set. The contents of the Greengenes database is skewed towards bacteria and contains less than 2% archeae and eukaryote sequences. The core tree also has more bacteria and may inflate the accuracy rates when testing against a mostly bacteria test taxa, as compared to the greater diversity of environmental samples. A taxon may be included in more than one test set and may be included in the overall results multiple times.

Current sequencing methods differ by the length of the read and cost per read. The Sanger method produces longer reads, up to 1500 nucleotides, but at a much higher cost. The 454 sequencing method allows for less expensive reads, but sacrifices length, producing reads of 300-400 nucleotides. The Illumina solution is the least expensive and has the shortest reads of up to 75 nucleotides. The SSU rRNA sequence is approximately 1500 nucleotides, making Sanger the best choice, but SSU rRNA has both highly conserved and more variable regions [35]. These regions are spaced out along the sequence. The region spanning from variable region V1 to V2 is approximately 300 nucleotides long. If the primers for PCR are selected to start replication from the variable regions, the reads can span the variable and intervening conserved regions. It has been shown that the short sequences spanning the variable regions are sufficient to determine the species [36].

One additional test set of 1000 sequences was created to simulate short reads. The nucleotides on each end of the randomly selected sequences are changed to gaps to reduce the sequence to range from V1 to V2, about 260 nucleotides. The short sequences are still aligned, can be compared against the full length sequences, and require no changes to the ParsInsert algorithm or comparison function.

5.2.2. Tree Insertion Accuracy

There are two results that must be tested for each inserted sequence. The first is whether insertion occurs at the correct location within the tree and the second is if the assigned taxonomy is accurate. Testing was accomplished in both cases by evaluating the taxonomy assigned to inserted sequences as compared to the curated taxonomy. In Table 3 two different types of matching are recorded: EXACT means the two taxonomies contain the same number of assigned ranks and all assignments match, and SUBSET implies that one taxonomy has fewer ranks assigned than the other (a subset), but all common ranks match. Table 3 displays the number of sequences in a test set, the number of exact taxonomy matches, the number of susbset matches, and the number of sequences that had at least one rank with a difference. The test used two full length sets (1000 and 5000) and one simulated short sequence set (1000).

Table 3: Test results of insertion points in the core tree by comparison of taxonomy.

	Number of sequences with known taxonomy	Exact match	Subset	Differences
Full length sequence (~1500 nucleotides)	997	624 (62%)	365 (37%)	8 (1%)
Full length sequence (~1500 nucleotides)	4947	2606 (53%)	2279 (46%)	42 (1%)
Short sequences (~240 nucleotides)	949	651 (68%)	249 (26%)	49 (5%)

Table 4 shows a breakdown of the accuracy as a function of the sequence similarity for the larger full length test set. The table shows that higher similarity to a known taxon or common ancestor leads to higher accuracy of the assigned taxonomy.

Table 4: Test results of insertion points in core tree by comparison of taxonomy, based on similarity of sequence to insertion point. No insertion points were assigned with less than 80% similarity.

5000 Full length sequences						
Sequence similarity	Total		Exact		Subset	
≥ 97%	3475	75%	3438	99%	12	0%
≥ 95%	524	11%	491	94%	10	2%
≥ 90%	525	1%	428	83%	29	6%
≥ 85%	99	2%	54	55%	22	22%
≥ 80%	8	0%	4	63%	2	25%

5.2.3. Taxonomy Accuracy

The method used to determine taxonomy accuracy compared the taxonomies at each rank. Taxonomies were only compared on the common levels of assigned taxonomy. There was no penalty for assignments fewer or more ranks than the reference assignment as there is no way to judge the additional information. The accuracy reported in Table 5 shows the number of assignments at each rank that are correct, divided by the number of assignments of that rank.

The algorithm performed extremely well, assigning 88% of the sequences to at least the Family rank, with full taxonomy accuracy of 99.4%. Assignment down to Genus rank occurred in 59% of the sequences, but only 12% of the sequences could be assigned a Species rank. Many of the sequences do not have taxonomies specified to the species rank, limiting the number of assignments that can be verified at that level. The accuracy for assignments at all ranks remained high, at better than 98%, showing that users can have high confidence in the assigned taxonomies.

Table 5: Overall taxonomy results.

Rank	Number of Assignments	% Assigned	Correct Assignments	% Correct
Kingdom	15546	100%	15546	100.0%
Phylum	15409	99%	15374	99.8%
Class	15305	98%	15247	99.6%
Order	14259	92%	14207	99.6%
Family	13601	88%	13525	99.4%
Genus	8939	58%	8773	98.1%
Species	1889	12%	1877	99.4%

ParsInsert assigns a confidence level of its assignment of a taxonomy based on the sequence similarity to the composite sequence at the insertion point. The accuracy levels are broken out in Table 6, showing that ParsInsert accurately predicts that the low accuracy taxonomy assignments it has made, come from the sequences that have a low confidence insertion points.

Table 6: Taxonomy assignment accuracy based on confidence level.

Rank	High Confidence			Medium Confidence			Low Confidence		
	Assigned	Correct	%	Assigned	Correct	%	Assigned	Correct	%
Kingdom	12317	12317	100.0%	2380	2380	100.0%	849	849	100.0%
Phylum	12298	12292	100.0%	2378	2370	99.7%	733	712	97.1%
Class	12267	12254	99.9%	2347	2330	99.3%	691	663	95.9%
Order	11725	11709	99.9%	2049	2032	99.2%	485	466	96.1%
Family	11480	11454	99.8%	1794	1771	98.7%	327	300	91.7%
Genus	8104	7970	98.3%	754	734	97.3%	81	69	85.2%
Species	1793	1786	99.6%	81	78	96.3%	15	13	86.7%

6. Phylogenetic Tree Visualization

The large number of sequences being processed by phylogenetic analysis has compounded the visualization problem for trees. Information overload required creation of phylogenetic tree images that can focus attention on the user's research points. I have designed and implemented PTreeView, a user-friendly application to visualize and customize phylogenetic trees. It has a broad range of functions and capabilities, such as formatting, zooming, searching, clade support with topology manipulation, bootstrap support, custom annotations, and support for large trees containing hundreds of thousands of taxa.

When trees were small, containing less than a few hundred taxa, visualization could be done using text graphics to create denagrams with taxa names and taxonomy annotations. As the number of sequences being analyzed continued to grow, the trees did not fit on a screen and applications to support efficient viewing of large trees became necessary. PTreeView is one of the few applications that can support the viewing of trees that contain the hundreds of thousands of sequences commonly available from current sequencing technologies.

6.1. Large Data Set Support (100,000s of Taxa)

The first visualization applications were developed to support viewing a few taxa and then a few hundred taxa. Today's sequencing methods produce hundreds of thousands to millions of reads per day. New methods of visualization and mechanisms for storing large trees are needed. Navigation features become necessary to manage the large amount of available data, such as searching for taxa of interest when less than one hundred taxa can be displayed at a time on the screen. Too much information is available and must be easily

grouped together, requiring a means of hiding detail until needed. Researchers must be able to manipulate the display of the tree, adding additional annotation and visual cues, such as color and text fonts, to show the results of their research.

6.2. Formatting (Color, Size, Font, …)

There are many features implemented in PTreeView that are not in other similar applications, such as node ordering, attribute display, import/export, searching functions, access to related web resources, and formatting of each node of the tree for color, size, and other attributes.

PTreeView supports reading tree files in common Newick or its own PTREEVIEW formats (see PTREEVIEW Format below, 6.8). The PTREEVIEW format maintains all visual states and extended attributes. Legends are automatically generated which show the scale of evolutionary distances represented by the length of branches. There are default preferences for colors, fonts, sizes, and alternate annotations to be displayed that can be set for the entire tree. Each tree can define titles, comments to be displayed at top and bottom, and other attributes of the tree as shown in Figure 6.

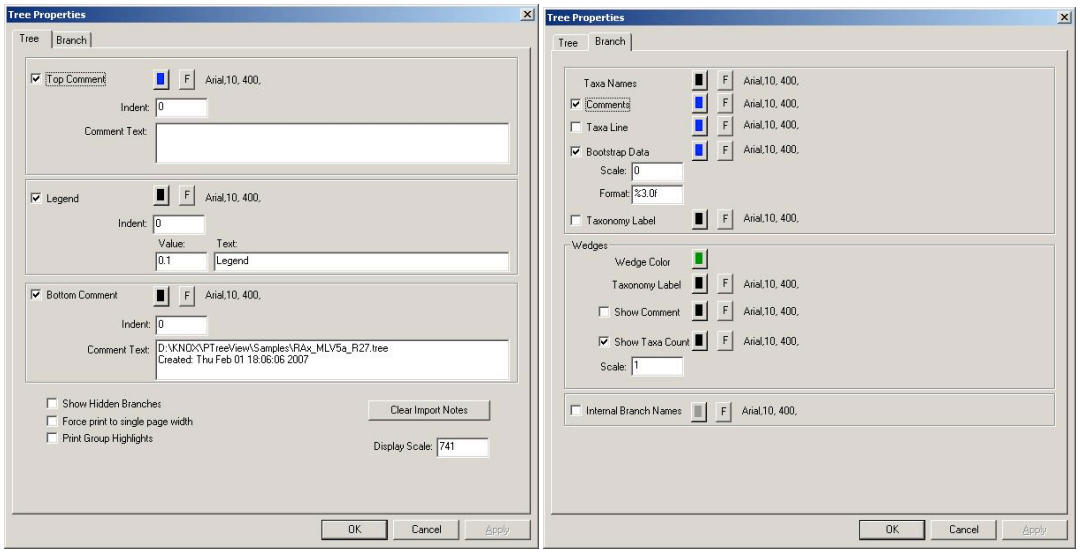

Figure 6 - Tree properties are applied to all items in the tree. Individual nodes can override the global settings with local settings for color, font, size, and display features.

Each individual branch or leaf can override the global settings. This allows an individual leaf or branch to be highlighted by changing color, font, or sizing of the text.

6.3. Topology Manipulation

Common navigational tools are available to zoom and pan the display through dragging a selection rectangle or using the CTRL+ and CTRL- keystrokes. The application will automatically jump and zoom to appropriate levels when searching.

6.3.1. Clades

The visualization of large trees requires methods of hiding parts of the information, in this case branches of the tree, to allow analysis of the data. This is accomplished by allowing the topology of the tree to be manipulated for display. Double clicking on a branch will cause the branch to collapse into a single object called a clade. The visual display of a clade is representational of the information within the cluster. Figure 7 shows a mixture of clades and taxa. The clades are represented by a trapezoid where the top and bottom line widths represent the maximum and minimum total branch lengths to taxa hidden within the

clade. The height of the clade represents the number of taxa within the clade. Double clicking on a clade to expand the clade will show the children in their previous state (collapsed or expanded).

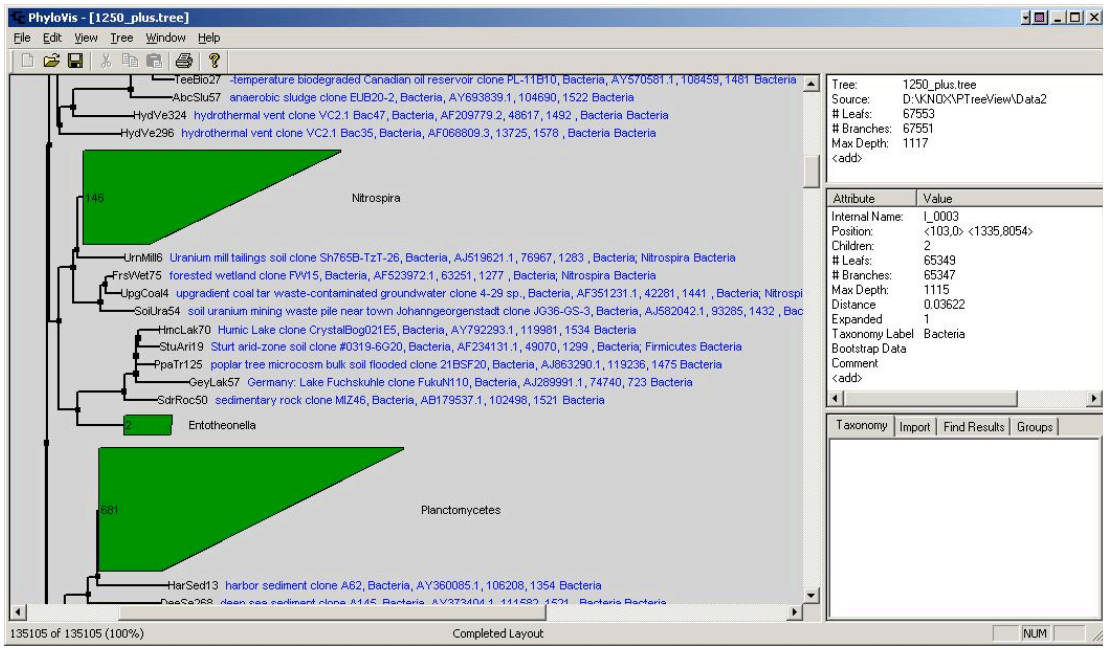

Figure 7 - Clade size represents the number of taxa contained within a clade. Lengths of the top and bottom represent the longest and shortest branch lengths contained in the clade.

6.3.2. Hiding Detail and Reordering

Changing the order in which the children of a branch are shown does not change the topology, but may make the tree more readable. PTreeView can sort the children from shortest to longest branch length, longest to shortest branch length, or by selecting a specific child and moving it up or down among the siblings. When a researcher is creating a representation of the tree for publication, they often need to hide portions of the tree to draw attention to other interesting features. PTreeView can mark a branch as “hidden” and it will not be displayed unless viewing of hidden items is selected for the entire tree.

6.3.3. Collapsing Branches

Another topological manipulation feature is creating a multi-furcating tree instead of a bifurcating tree. There are often sets of taxa where the information needed for predicting common ancestry between them is lacking. For example, given taxa a, b, and c, there are three ways these could have evolved, a-b common ancestor before a-b-c ancestor, a-c ancestor before a-b-c, or c-b ancestor before a-b-c. If the data does not support one of these more than the others, the tree can have three children of the a-b-c common ancestor. The support level for a given common ancestor is given by a bootstrap value calculated from the set of possible tree topologies (see Bootstrap Values section 6.4.3). When support for a common ancestor falls below a given value, the topology can be collapsed to reflect this information. PTreeView supports manual manipulation of the tree, moving all children into a grandparent, or performing a complete tree analysis and automatically collapsing all branches where the support level is below a given value.

6.3.4. Orientation

A unique feature available in PTreeView is the display of the tree in right-to-left and left-to right orientations shown in Figure 8. This is valuable when comparing two trees since it places taxa towards each other. Search functions allow highlighting of nodes simultaneously in multiple trees.

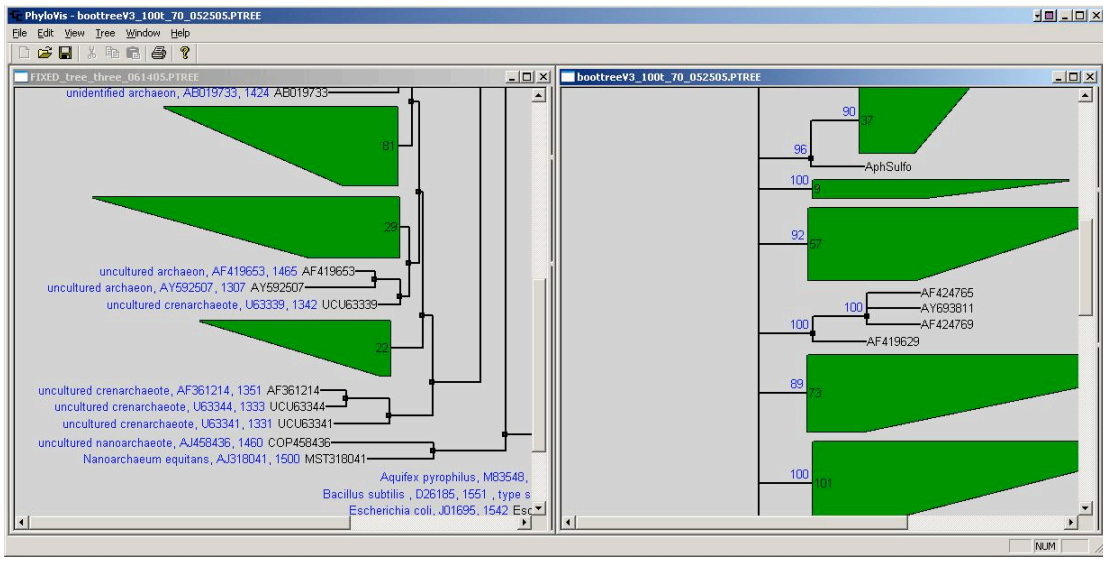

Figure 8 - Left-to-Right and Right-to-Left orientation features of PTreeView allow trees to be easily compared.

The trees created by phylogenetic analysis are sometimes not rooted at any point in the tree. This means that the tree can be picked up at any branching point in the tree, allowing the selected branch to become the root, and insuring all other branch lengths and topology remain the same. PTreeView implements this feature by selecting a root node and reshaping the tree while conserving all branch lengths between taxa.

6.4. Custom Annotation

There are many attributes that PTreeView has predefined for use in creating and maintaining the display environment. There are also attributes that can be imported into PTreeView from external sources. Annotations can be collected from external databases and applied to the current tree.

6.4.1. Comments

Every node of the tree can have a comment. The comments are optionally displayed by user control. The information contained is user defined, but usually gives detail as to the source of the sample and/or information about the taxonomy. Comment values

for clades are usually taxonomy and can be optionally displayed either in collapsed or expanded states.

6.4.2. Distances

Distances are stored as application attributes for each node of the tree. The user can manipulate these distances. The attributes can be edited directly or imported from external sources. Many times the researcher has trees that are not fully annotated with branch lengths. PTreeView can also read the distances from another tree or data file.

6.4.3. Bootstrap Values

Bootstrapping is a statistical method of estimating the probability of each subtree of the tree. Because many of the tree building algorithms are dependent on taxa input order, the distribution of the possible trees is determined by running the algorithm a number of times with random input ordering. To create bootstrap values for a given tree, each common ancestor is evaluated to determine how often that cluster is found in the generated trees. The bootstrap value for any cluster is the percentage of generated trees that have that cluster. Therefore, when a specific grouping of taxa is always clustered together, the bootstrap value is 100% for that group's common ancestor.

PTreeView can annotate a tree by importing bootstrap values. There are many applications available which generate bootstrap values. They usually create a Newick format tree file where the distances are actually the bootstrap values. PTreeView imports the tree and then searches the current tree for matches to clusters from the import tree. When a match is found, a bootstrap value is set for that common ancestor node.

PTreeView also has the capability to read a set of trees from which to calculate the bootstrap support level explicitly. The current tree is searched for clusters from each of the bootstrap trees. A counter at each common ancestor in the tree is incremented when an exact descendant list is found in a bootstrap tree. Dividing the counter by the number of trees processed gives the percentage of trees supporting that ancestor, which is the bootstrap value.

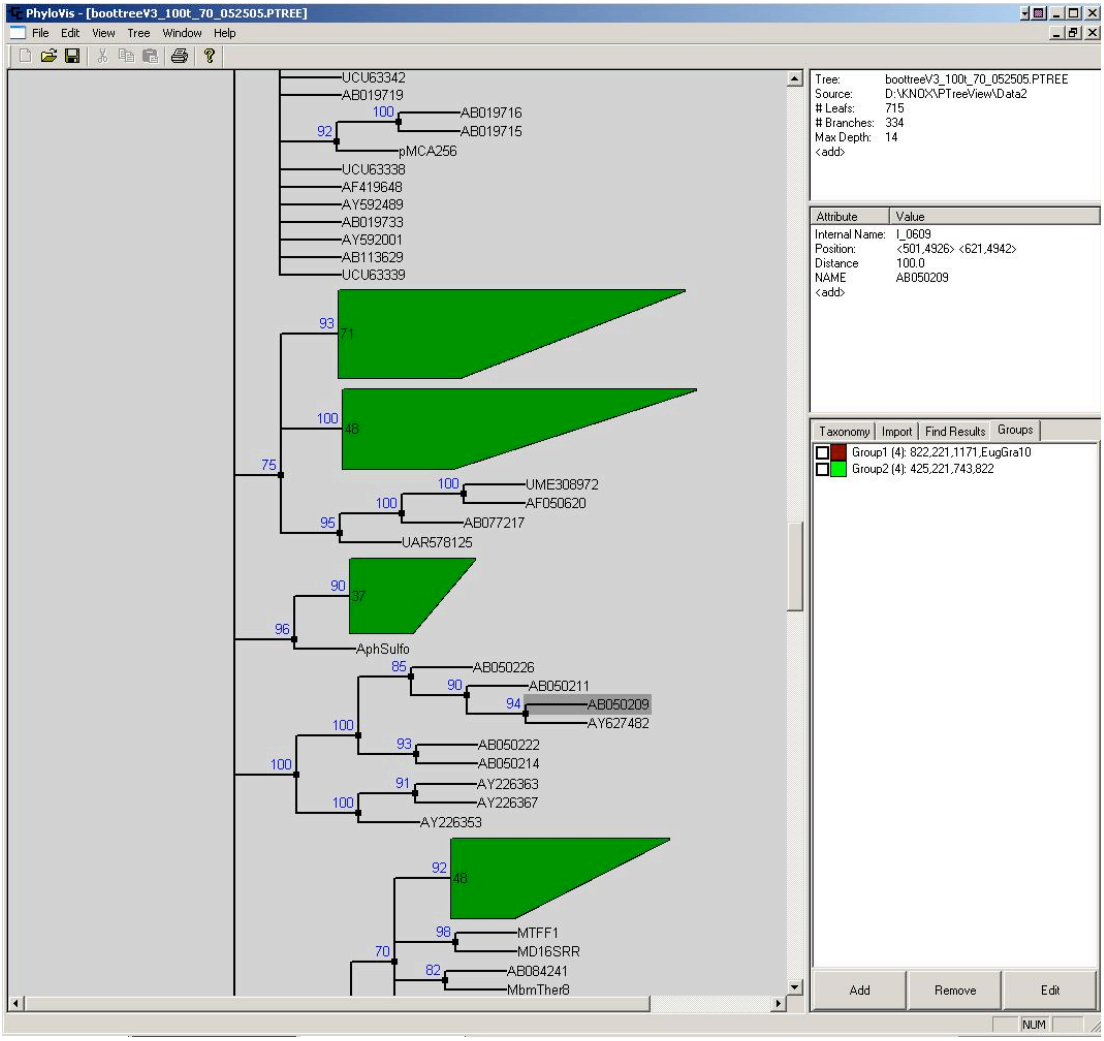

Figure 9 - Bootstrap values are displayed as numbers to the left of the branch point. PTreeView supports functions to modify the topology by collapsing branches that are below a given bootstrap value. The bootstrap values can be imported from other trees or the support levels can be calculated by evaluating a set of bootstrap trees.

6.4.4. User Defined Attributes

Users often have more information about the sample set which they are studying. This information is different in almost all instances, such as sex, location, and time of day for samples in human tests, or pH, nitrogen, and temperature for environmental samples. This information can be stored as part of the comments, but it may clutter the display. PTreeView allows the user to import and assign custom values to any node of the tree. The Search feature has the ability to find nodes based on user defined attributes.

6.5. Searching

6.5.1. Find

Once the trees become too large to be represented on a single screen, navigation to specific locations becomes more difficult. To aid the analysis of large trees, a sophisticated search method has been implemented. The search can find information within a specific field or a regular expression can be used to define more complex patterns (Figure 10). The search results are displayed in a results tab and clicking on a result item will jump and zoom the display to show that resulting node. All the results of the search are marked on the display with a marker. This marker is also shown on any clade containing a result item as shown in Figure 11.

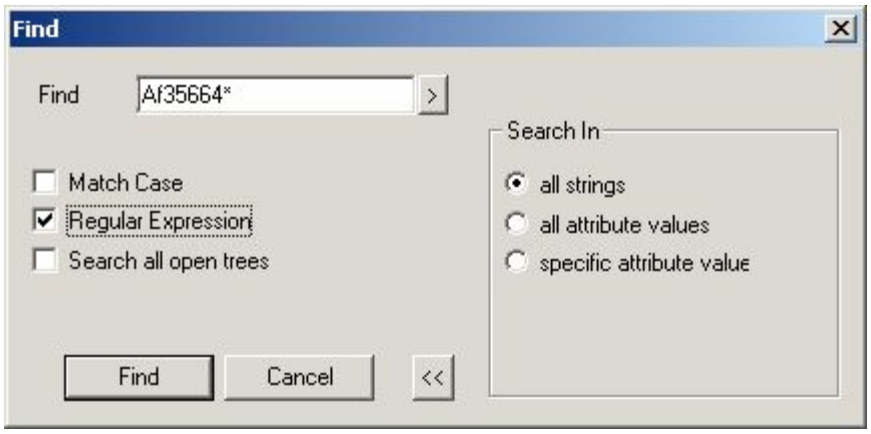

Figure 10 - Searching dialog handles simple string matching and regular expressions. Search can be limited to only attribute values or a specific attribute. The search function can be performed simultaneously in all open trees.

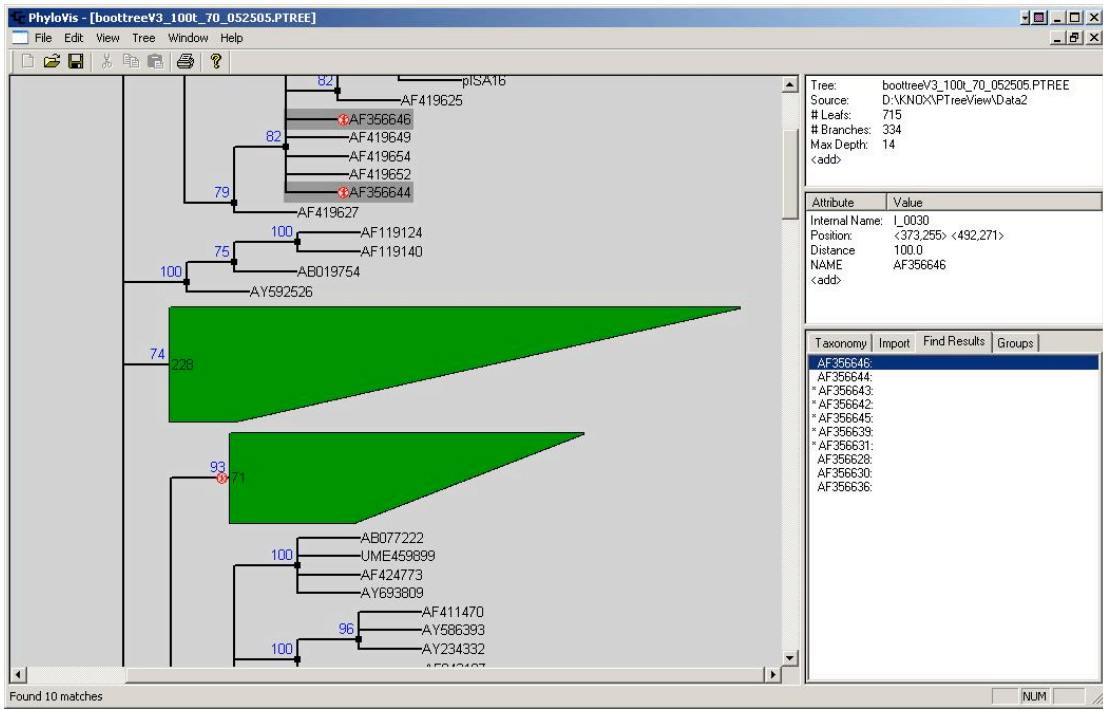

Figure 11 - Search results shown as a list in the lower right and by red marker highlighting individual taxa or clades in the tree display. Clicking on an item in the results automatically scrolls the display to bring the selected item into view.

6.5.2. Groups

Many users will be interested in a subset of the tree's taxa. Often they are researching a specific branch or set of taxa. PTreeView has implemented a group function,

which defines groups as a set of taxa that are stored system wide. They are read from an initialization file at startup. Groups can be created from a search result or copied from a set of taxa names. Each group can be assigned a color for marking the tree as seen in Figure 12. Once defined, the groups can be individually set to display a marker. For taxa belonging to multiple groups, all marks are visible. As with the search results, Figure 13 show that when a clade contains a member of a group, it is marked with the group's marker.

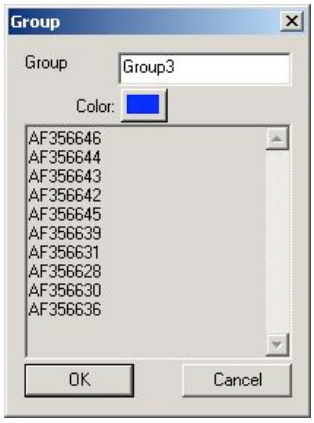

Figure 12 - PTreeView can define groups of taxa to be highlighted in the specified color. Groups are persistant over all trees.

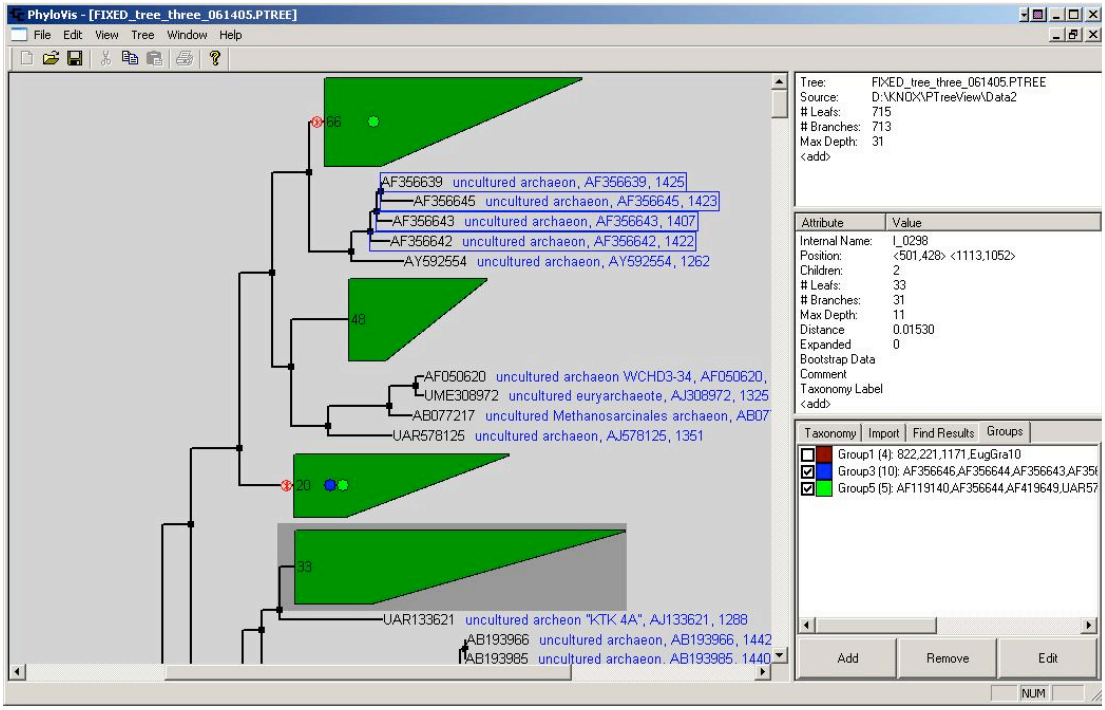

Figure 13 - Groups can be defined for sets of interesting taxa. The members of the groups are highlighted by drawing a rectangle around individual taxa or with a colored marker when a clade contains a member of the group. The Groups tab at the bottom right allows individual groups to be highlighted as needed.

6.6. Import / Export

Many applications are available which create and manipulate trees in the Newick format. Many cannot handle the long strings which are associated with a tree node when all attributes are stored as part of the Newick comment field. As a result, most applications just use a single id for the name of the taxa. Viewers of the tree would frequently like to see more information or annotation of the taxa, such as assigning the taxonomy as part of the comment field. This information is can be created as an auxiliary file and imported into PTreeView. PTreeView supports reading and writing comma separated value (CSV) files containing attributes for nodes within the tree. PTreeView can reconcile common ancestors of a set of taxa for assignment of attributes. It can also export a user defined set of

attributes from the whole or a subset of the tree. Features have been implemented for the special cases of importing taxonomy, distances, or bootstrap values from other trees.

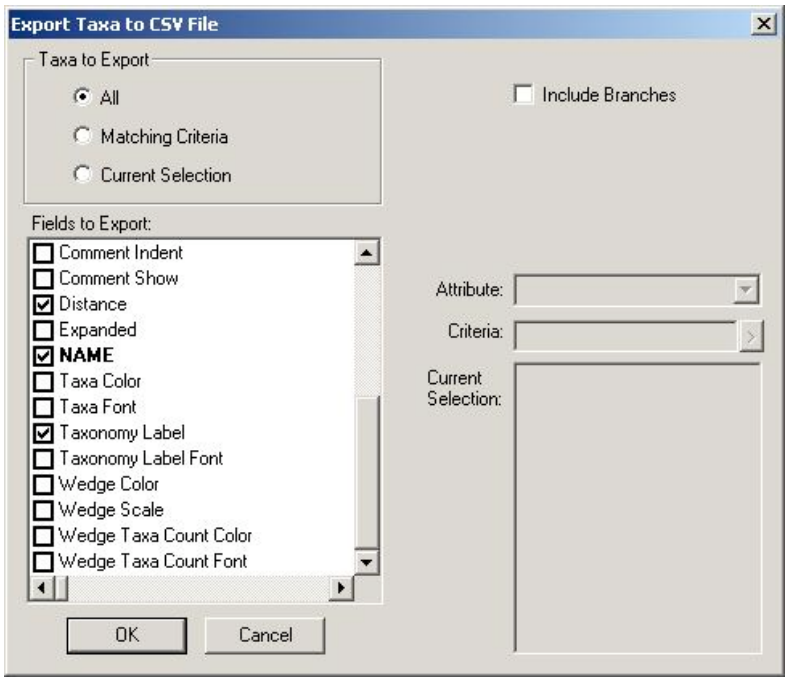

Figure 14 – The Export function lists all possible attributes, including user-defined attributes. Only the selected information is exported to the comma separated formatted file. The Import feature will read any header information from the file selected for import and displays a similar selection list for columns to import.

6.7. Access to Web Resources

While PTreeView can assign any attributes to individual taxa in the tree, usually there is far too much available data to be assigned within the tree. To allow users as much freedom as possible, an access method from a given taxa to standard databases was implemented. Most of the work performed and examples used in this thesis have been drawn from SSU rRNA research and the Greengenes Database has been used extensively. When examining the trees produced by SeqCluster or ParsInsert, access to information stored in the database records for that particular organism is required for further analysis. Right clicking on a taxon passes the current taxon name to a database of choice (Figure 15). Depending on the database interface, the database entry is either immediately displayed or

a search result page is displayed from which the data can be accessed (Figure 16). Filters were implemented to access the following databases: Greengenes (by prokMSA id, assession number), Silva, and NCBI (nucleotide, protein, genome, structure, taxonomy, pubmed) databases.

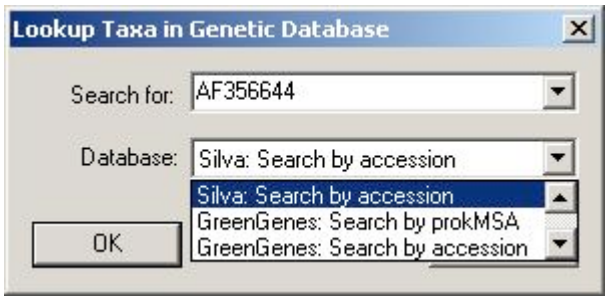

Figure 15 - Select database and type of search key to directly access database information.

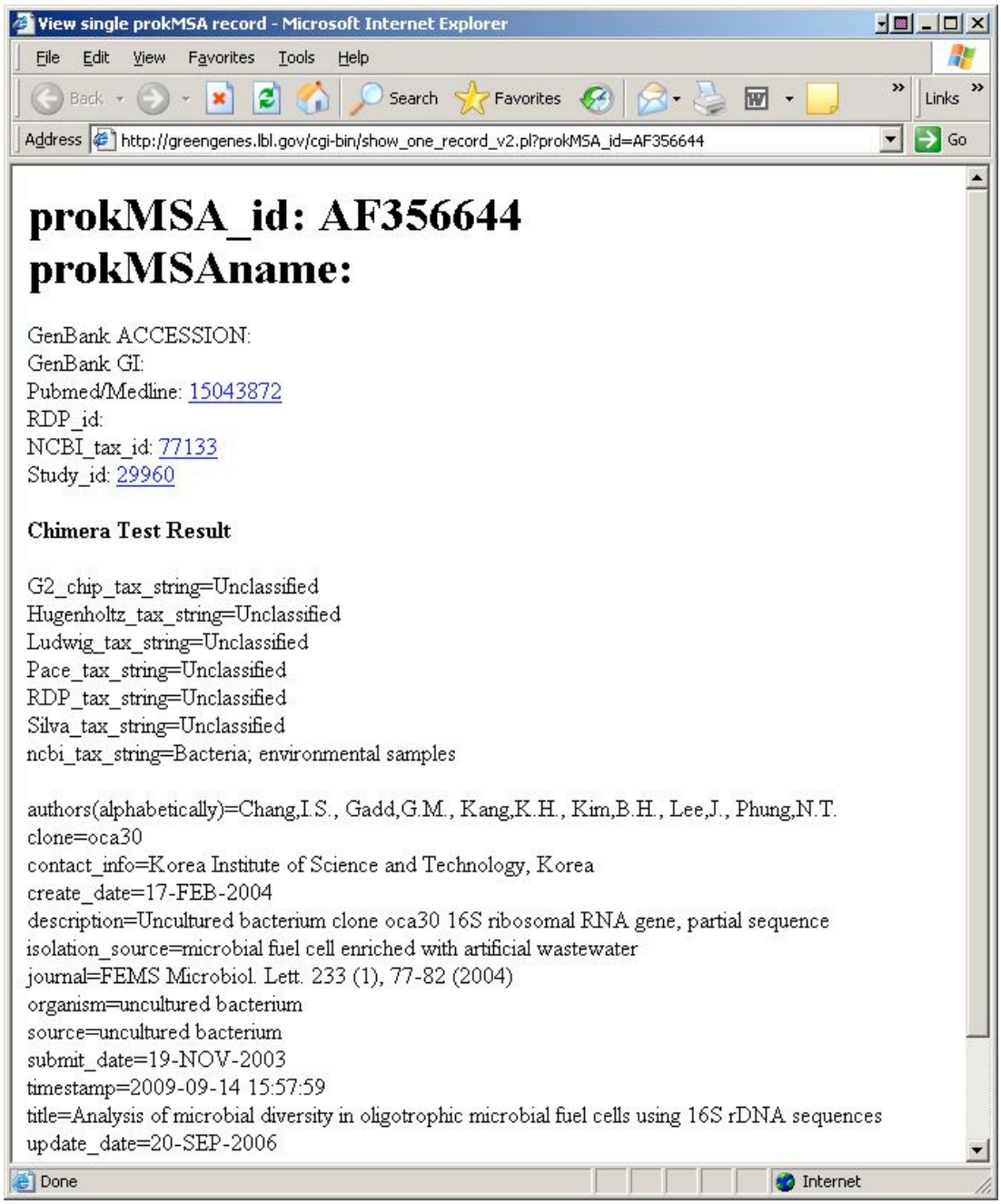

Figure 16 - Automated access directly to Greengenes database for selected taxon.

6.8. PTreeView File Format

The Newick Tree format is a simplistic format that does not easily allow for customization or extension. The format only defines fields for a name, a comment, and a

branch length for each node. All annotations must be placed in the comment field. Though other applications have used the comment field to store additional information, it quickly becomes unreadable and difficult to parse. To allow the additional attributes used by the visualizer, a new format was developed. XML is considered overly verbose and difficult to read, but does offer a simple method for extending the variety of stored information. A simple XML-like format that is easily parsed was created to support both the visualizer and end user annotations.

The PTREEVIEW file format is similar to an XML format, but eliminates some of the textual overhead by using a simple recursive definition (Figure 17). Each object has a type and a list of associated objects. Figure 18 shows an example of a simple tree. The PTREEVIEW tree format can support multiple tree definitions per file. Every '<' defines the start of an object and '>' terminates the current object definition. All objects defined within the brackets are assigned to the containing object. There are only three object types: TREE, which defines the top most node, NODE, which defines a subnode to the current node, and ATTR, which defines attributes of the current node.

```
# lines beginning with '#' are treated as comment lines
tree  := '<' TREE object_list '>' where TYPE is defined as TREE, NODE, or ATTR.
object_list :=  object | object object_list
object := attr_object | node_object
attr_object := '<' 'ATTR' name value '>'
node object := '<' 'NODE'  object_list '>'
```

Figure 17 - PTreeView Format Specification can describe the tree using only three object types: TREE, NODE, and ATTR (attribute). A simple recursive parser easily parses the format. The format can support any number of named attributes, which can be passed on without interpretation by applications not using a particular attribute, which also mediates problems with down-level application versions.

The file format allows any information to be stored into the nodes. The visualizer uses this format to maintain current visual attributes such as color, font, and size

information. It is also used to store user defined attributes and values. These attributes are conserved between sessions of the application.

```
<TREE
        <ATTR “Bottom Comment” “This is a comment displayed at bottom of tree”>
        <NODE
                <ATTR “Name” “root”>
                <NODE
                        <ATTR “Name” “20394”>
                        <ATTR “Distance” “0.00456”>
                >
                <NODE
                        <ATTR “Name” “8345783”>
                        <ATTR “Distance” “0.0126”>
                >
        >
>
```

Figure 18 - Simple example of PTreeView format for storing attributes and topology of the tree.

Figure 18 shows the definition of a tree with only a root node and two child nodes. The names of the nodes and the distance from the parent are the only defined attributes. The format is simple to read, easily editable, and highly extendable. Attributes that are not recognized or used by an application are ignored and transferred to the output tree without the need to understand the formatting. Figure 19 shows an example of a tree with many attributes and nodes, including color, font, display options for the whole tree, and bootstrap values for common ancestor nodes.

```
<TREE
<ATTR "Legend Color#" "0x00000000"> <ATTR "Taxonomy Label Font!" ""> <ATTR "Bootstrap Show~" "1"> <ATTR "Text Font!" "">
<ATTR "Legend Font!" ""> <ATTR "Display Scale" "0.060287"> <ATTR "Taxa Line Show~" "0"> <ATTR "Caption Show~" "1"> <ATTR
"Comment Font!" ""> <ATTR "Branch Name Font!" ""> <ATTR "End Comments Show~" "1"> <ATTR "Clade Size Show~" "1"> <ATTR
"Legend Text" "Legend"> <ATTR "Clade Font!" ""> <ATTR "Caption Color#" "0x00000000"> <ATTR "Taxa Line Color#" "0x00000000">
<ATTR "Taxonomy Label Color#" "0x00000000"> <ATTR "Bootstrap Font!" ""> <ATTR "Clade Scale" "1"> <ATTR "Caption Font!" "">
<ATTR "Legend Show~" "1"> <ATTR "Taxa Line Font!" ""> <ATTR "Legend Value" "0.100000"> <ATTR "Comment Color#"
"0x00000000"> <ATTR "Bootstrap Color#" "0x00FF0000"> <ATTR "Text Color#" "0x00000000"> <ATTR "Comments Show~" "1">
<ATTR "Caption" "C:\\GN\\042507\\ptree\\sample.tree
Created: Wed Apr 25 14:22:43 2007"> <ATTR "Caption Indent" "0"> <ATTR "Bootstrap Format" "%3.0f"> <ATTR "Rooted" "9741">
...
 <NODE <ATTR "Distance" "249.0"> <ATTR "Bootstrap Data" "249.000000">
    <NODE <ATTR "Distance" "300.0"> <ATTR "Bootstrap Data" "300.000000">
      <NODE <ATTR "Distance" "300.0"> <ATTR "Bootstrap Data" "300.000000">
          <NODE <ATTR "Distance" "300.0"> <ATTR "Bootstrap Data" "300.000000">
             <NODE <ATTR "Comment" " 1440 , EF069377.1, Archaea; Crenarchaeota; Thermoprotei; marine archaeal group 1;”>
                    <ATTR "Distance" "300.0"> <ATTR "NAME" "195536">
              >
              <NODE <ATTR "Comment" " 1440 , EF069368.1, Archaea; Crenarchaeota; Thermoprotei; marine archaeal group 1; “>
                     <ATTR "Distance" "300.0"> <ATTR "NAME" "178096">
              >
            >
            <NODE <ATTR "Comment" " 1440 , EF069337.1, Archaea; Crenarchaeota; Thermoprotei; marine archaeal group 1; ">
                    <ATTR "Distance" "300.0"> <ATTR "NAME" "188374">
            >
          >
          <NODE <ATTR "Comment" " 1279 , AY591933.1, Archaea; environmental samples">
                  <ATTR "Distance" "300.0"> <ATTR "NAME" "104937">
        >
      >
...
```

Figure 19 - Complex set of attributes including font, color, and display state information for a tree. The PTREEVIEW file format is easy to read, supports both application and user attributes per node, and is simple to parse. The fields described in the Newick tree format, Name, Distance and Comment, are stored as attributes.

7. Conclusions and Future Work

This thesis has described innovative solutions to support researchers exploring bacterial communities. The applications shown here directly address the current computational bottlenecks faced everyday by researchers. As the size of data sets keeps increasing and the sequencing technology keeps improving the length and quantity of reads per run, better taxonomic and phylogenetic analysis algorithms will be needed to allow timely analysis. Applications, such as SeqCluster, ParsInsert, and PTreeView, will enable researchers to find the relationships between communities and discover the science behind the larger data sets, which will ultimately lead to better solutions to the problems of pollution and disease.

7.1. Future Work: Clustering Data Sets

7.1.1. Nondeterministic Selection of Representative Sequence

The clustering algorithm should be able to randomly select a representative from the cluster based on its distance to the cluster center. Currently, only the sequence closest to the center is selected as the representative. Other sequences that are close to the cluster center should have a chance to be selected as the representative sequence, but become less likely as the distance from the center increases. This stochastic method is important when building trees for bootstrapping purposes, as the alternate sequences may give different resulting trees.

7.1.2. Benchmark Speed and Accuracy Against Other Solutions

The SeqCluster application can handle more sequences than most other solutions, but it must be benchmarked against other popular solutions to obtain wider acceptance of the application.

7.2. Future Work: Phylogenetic Tree and Taxonomy Classification

7.2.1. Evolutionary Molecular Models

The algorithms used for sequence comparison and calculating evolutionary distance assume a simple molecular clock. There are better models that use more complex clock models to determine the evolutionary time between two sequences. These should give more accurate results for determining insertion locations and distances from common ancestors.

7.2.2. Support Multiple Alignments per Sequence

Phylogenetic tree building is dependant on the alignment of the unknown sequences to the set of core sequences. The alignment process will often determine a small set of good possible alignments for each sequence, but only one is selected for output. The best alignment picked by a scoring function is sometimes not the one picked by an expert researcher. All of the good alternate alignments could be used as weighted inputs to ParsInsert. The time, t, required for determining the insertion point is constant for a given core tree. If there were s alignments for each unknown sequence, the total time for each insertion would increase to s*t. The added benefit would be an increase in the overall accuracy of locating the correct insertion points. The different sequence alignments may lead to multiple areas within the tree as possible insertion points, which would lower the confidence of the insertion point and could be resolved by selecting the highest overall

match. The sequence could also be marked as having alternative taxonomy to allow further study. A small adjustment to the PasrInsert algorithm would involve comparing all of the sequence alignments of the unknown sequence and adjusting the score using the alignment weighting.

7.3. Future Work: Visualization

7.3.1. Output Types

Current graphical output of the tree viewer is limited to postscript files, which allows manipulation within Adobe Acrobat for final clean up before publishing. The application should support the common graphical formats (GIF, JPEG, TIFF) and the PDF format.

7.3.2. Platform Independent Solution

PTreeView's current implementation is only available on the Microsoft Windows platform. The user interface and visualization routines were designed and written for the Windows Graphical Interface Libraries and would require redesign for another platform's Graphical User Interface.

8. References

[1] N. R. Pace, "A molecular view of microbial diversity and the biosphere," *Science,* vol. 276, pp. 734-40, May 2 1997.

[2] J. B. Hughes, J. J. Hellmann, T. H. Ricketts, and B. J. Bohannan, "Counting the uncountable: statistical approaches to estimating microbial diversity," *Appl Environ Microbiol,* vol. 67, pp. 4399-406, Oct 2001.

[3] A. Chao, W. H. Hwang, Y. C. Chen, and C. Y. Kuo, "Estimating the number of shared species in two communities," *Statistica Sinica,* vol. 10, pp. 227-246, Jan 2000.

[4] A. E. McCaig, L. A. Glover, and J. I. Prosser, "Molecular analysis of bacterial community structure and diversity in unimproved and improved upland grass pastures," *Appl Environ Microbiol,* vol. 65, pp. 1721-30, Apr 1999.

[5] T. Z. DeSantis, P. Hugenholtz, N. Larsen, M. Rojas, E. L. Brodie, K. Keller, T. Huber, D. Dalevi, P. Hu, and G. L. Andersen, "Greengenes, a chimera-checked 16S rRNA gene database and workbench compatible with ARB," *Appl Environ Microbiol,* vol. 72, pp. 5069-72, Jul 2006.

[6] J. Felsenstein, "Inferring phylogenies from protein sequences by parsimony, distance, and likelihood methods," *Methods Enzymol,* vol. 266, pp. 418-27, 1996.

[7] Z. Liu, T. Z. DeSantis, G. L. Andersen, and R. Knight, "Accurate taxonomy assignments from 16S rRNA sequences produced by highly parallel pyrosequencers," *Nucleic Acids Res,* vol. 36, p. e120, Oct 2008.

[8] L. Krause, N. N. Diaz, A. Goesmann, S. Kelley, T. W. Nattkemper, F. Rohwer, R. A. Edwards, and J. Stoye, "Phylogenetic classification of short environmental DNA fragments," *Nucleic Acids Res,* vol. 36, pp. 2230-9, Apr 2008.

[9] S. Tzahor, D. Man-Aharonovich, B. C. Kirkup, T. Yogev, I. Berman-Frank, M. F. Polz, O. Beja, and Y. Mandel-Gutfreund, "A supervised learning approach for taxonomic classification of core-photosystem-II genes and transcripts in the marine environment," *BMC Genomics,* vol. 10, p. 229, 2009.

[10] J. R. Cole, B. Chai, R. J. Farris, Q. Wang, S. A. Kulam, D. M. McGarrell, G. M. Garrity, and J. M. Tiedje, "The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis," *Nucleic Acids Res,* vol. 33, pp. D294-6, Jan 1 2005.

[11] P. D. Schloss and J. Handelsman, "Introducing DOTUR, a computer program for defining operational taxonomic units and estimating species richness," *Appl Environ Microbiol,* vol. 71, pp. 1501-6, Mar 2005.

[12] T. Pommier, B. Canback, P. Lundberg, A. Hagstrom, and A. Tunlid, "RAMI: a tool for identification and characterization of phylogenetic clusters in microbial communities," *Bioinformatics,* vol. 25, pp. 736-742, Mar 15 2009.

[13] W. H. E. Day and H. Edelsbrunner, "Efficient Algorithms for Agglomerative Hierarchical-Clustering Methods," *Journal of Classification,* vol. 1, pp. 7-24, 1984.

[14] I. Elias and J. Lagergren, "Fast computation of distance estimators," *BMC Bioinformatics,* vol. 8, pp. -, Mar 13 2007.

[15] D. Dalevi, T. Z. Desantis, J. Fredslund, G. L. Andersen, V. M. Markowitz, and P. Hugenholtz, "Automated group assignment in large phylogenetic trees using GRUNT: GRouping, Ungrouping, Naming Tool," *BMC Bioinformatics,* vol. 8, p. 402, 2007.

[16] R. Knight, P. Maxwell, A. Birmingham, J. Carnes, J. G. Caporaso, B. C. Easton, M. Eaton, M. Hamady, H. Lindsay, Z. Z. Liu, C. Lozupone, D. McDonald, M. Robeson, R. Sammut, S. Smit, M. J. Wakefield, J. Widmann, S. Wikman, S. Wilson, H. Ying, and G. A. Huttley, "PyCogent: a toolkit for making sense from sequence," *Genome Biology,* vol. 8, pp. -, 2007.

[17] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. R. Gilbert, I. Korf, H. Lapp, H. Lehvaslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, and E. Birney, "The bioperl toolkit: Perl modules for the life sciences," *Genome Research,* vol. 12, pp. 1611-1618, Oct 2002.

[18] T. Z. DeSantis, Jr., P. Hugenholtz, K. Keller, E. L. Brodie, N. Larsen, Y. M. Piceno, R. Phan, and G. L. Andersen, "NAST: a multiple sequence alignment server for comparative analysis of 16S rRNA genes," *Nucleic Acids Res,* vol. 34, pp. W394-9, Jul 1 2006.

[19] E. Pruesse, C. Quast, K. Knittel, B. M. Fuchs, W. Ludwig, J. Peplies, and F. O. Glockner, "SILVA: a comprehensive online resource for quality checked and aligned ribosomal RNA sequence data compatible with ARB," *Nucleic Acids Res,* vol. 35, pp. 7188-96, 2007.

[20] J. R. Cole, B. Chai, T. L. Marsh, R. J. Farris, Q. Wang, S. A. Kulam, S. Chandra, D. M. McGarrell, T. M. Schmidt, G. M. Garrity, and J. M. Tiedje, "The Ribosomal Database Project (RDP-II): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy," *Nucleic Acids Res,* vol. 31, pp. 442-3, Jan 1 2003.

[21] W. Ludwig, O. Strunk, R. Westram, L. Richter, H. Meier, Yadhukumar, A. Buchner, T. Lai, S. Steppi, G. Jobb, W. Forster, I. Brettske, S. Gerber, A. W. Ginhart, O. Gross, S. Grumann, S. Hermann, R. Jost, A. Konig, T. Liss, R. Lussmann, M. May, B. Nonhoff, B. Reichel, R. Strehlow, A. Stamatakis, N. Stuckmann, A. Vilbig, M. Lenke, T. Ludwig, A. Bode, and K. H. Schleifer, "ARB: a software environment for sequence data," *Nucleic Acids Res,* vol. 32, pp. 1363-71, 2004.

[22] W. M. Fitch, "Distinguishing homologous from analogous proteins," *Syst Zool,* vol. 19, pp. 99-113, Jun 1970.

[23] F. Ronquist, "Fast Fitch-parsimony algorithms for large data sets," *Cladistics-the International Journal of the Willi Hennig Society,* vol. 14, pp. 387-400, Dec 1998.

[24] P. A. Goloboff, "Analyzing large data sets in reasonable times: Solutions for composite optima," *Cladistics-the International Journal of the Willi Hennig Society,* vol. 15, pp. 415-428, Dec 1999.

[25] P. A. Goloboff and D. Pol, "On divide-and-conquer strategies for parsimony analysis of large data sets: Rec-I-DCM3 versus TNT," *Systematic Biology,* vol. 56, pp. 485-495, 2007.

[26] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster, "MEGAN analysis of metagenomic data," *Genome Res,* vol. 17, pp. 377-86, Mar 2007.

[27] D. H. Huson, D. C. Richter, C. Rausch, T. Dezulian, M. Franz, and R. Rupp, "Dendroscope: An interactive viewer for large phylogenetic trees," *BMC Bioinformatics,* vol. 8, p. 460, 2007.

[28] J. Bingham and S. Sudarsanam, "Visualizing large hierarchical clusters in hyperbolic space," *Bioinformatics,* vol. 16, pp. 660-1, Jul 2000.

[29] S. Kumar, K. Tamura, and M. Nei, "MEGA3: Integrated software for Molecular Evolutionary Genetics Analysis and sequence alignment," *Brief Bioinform,* vol. 5, pp. 150-63, Jun 2004.

[30] J. Felsenstein, "PHYLIP (PHYLogeny Inference Package) version 3.66."

[31] F. Chevenet, C. Brun, A. L. Banuls, B. Jacq, and R. Christen, "TreeDyn: towards dynamic graphics and annotations for analyses of trees," *BMC Bioinformatics,* vol. 7, p. 439, 2006.

[32] T. Munzner, Fran\, \#231, o. Guimbreti\, \#232, re, S. Tasiran, L. Zhang, and Y. Zhou, "TreeJuxtaposer: scalable tree comparison using Focus+Context with guaranteed visibility," *ACM Trans. Graph.,* vol. 22, pp. 453-462, 2003.

[33] R. D. Page, "TreeView: an application to display phylogenetic trees on personal computers," *Comput Appl Biosci,* vol. 12, pp. 357-8, Aug 1996.

[34] C. A. Lozupone, M. Hamady, S. T. Kelley, and R. Knight, "Quantitative and qualitative beta diversity measures lead to different insights into factors that structure microbial communities," *Appl Environ Microbiol,* vol. 73, pp. 1576-85, Mar 2007.

[35] Y. Van de Peer, S. Chapelle, and R. De Wachter, "A quantitative map of nucleotide substitution rates in bacterial rRNA," *Nucleic Acids Res,* vol. 24, pp. 3381-91, Sep 1 1996.

[36] S. Chakravorty, D. Helb, M. Burday, N. Connell, and D. Alland, "A detailed analysis of 16S ribosomal RNA gene segments for the diagnosis of pathogenic bacteria," *J Microbiol Methods,* vol. 69, pp. 330-9, May 2007.