To the Graduate Council:

I am submitting herewith a thesis written by Shakhina Abdimajidovna Pulatova entitled “A Whole Genome Phylogeny Using Truncated Pivoted QR Decomposition.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Computer Science.

Michael W. Berry
Major Professor

We have read this thesis
and recommend its acceptance:

Robert C. Ward

Kwai L. Wong

Accepted for the Council:

Anne Mayhew
Vice Chancellor and
Dean of Graduate Studies

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

A Whole Genome Phylogeny Using Truncated Pivoted QR Decomposition

A Thesis
Presented for the
Master of Science Degree
The University of Tennessee, Knoxville

Shakhina Abdimajidovna Pulatova

August 2004

Dedication

To my dearest grandparents:

Habiba G'ulomqodir qizi and *Po'lat Mo'min*,

Sabura Fozilova and *Abduqayum Pulatov*

and to my loving parents:

Tabassum Muminova and *Abdimajid Pulatov*

Acknowledgements

I would like to extend my deepest gratitude to my advisor Dr. Michael Berry for giving me opportunity to work on this project, and for his guidance, motivation, sound advice and endless support throughout this research. Not only have I acquired experience in bioinformatics and numerical algebra, but also my writing skills have improved under his guidance. I also owe many thanks to Dr. Gary Stuart for his invaluable help in biological aspects of the project including answering my numerous questions regarding phylogeny. I am sincerely thankful for the help provided by Dr. G. W. Stewart which led to my better understanding of the algorithm used in this study. I am also grateful to Dr. Robert Ward and Dr. Kwai Wong for agreeing to serve on my committee. Thanks to Murray Browne for his patience in reading and improving the draft versions of this thesis. I wish to express my geniune appreciation to JICS people for providing funding throughout my Master's program, especially to Dr. Christian Halloy, Dr. Kwai Wong and Shawn Ericson for providing flexible schedules and an amiable working atmosphere.

Most importantly, I am extremely thankful to my family for always being there for me. I could never thank my loving parents enough for their understanding, patience, encouragement to always pursue my goals, and their absolute confidence in me. I sincerely appreciate their financial and moral support for my studying in the U.S.A. I am grateful to my siblings Farzona and Firdavs for their love and support.

Abstract

The increasing availability of whole genome sequences in public databases has stimulated the development of new methods to automatically compare and categorize genes and species. Recently developed methods based on the singular value decomposition (SVD) allow for the simultaneous identification and definition of well concerved motifs and gene families using very large whole genome datasets. In contrast, this work discusses the use of a truncated pivoted QR factorization as a scalable alternative to the SVD for comparing whole genomes in a phylogenetic context. This algorithm computes the R factor of the decomposition without forming the Q factor or altering the original matrix. Encodings for both proteins and peptides are obtained by applying the QR factorization to a large and sparse peptide-by-protein data matrix in two ways. Gene and species phylogenies comparable to those from the SVD approach are constructed using cosines as pairwise similarities of the resulting protein vectors. Performance evaluations conducted on a few genomic datasets indicate that such an approach presents an efficient alternative to the SVD-based method for whole genome phylogeny.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The amount of whole genome sequences in public databases such as GenBank[1] and Swiss-Prot[2] is expanding rapidly. To date, the entire human genome along with several model organisms have been sequenced and annotated, and the number and complexity of genomic sequences continues to advance. Based on the theory that sequence similarity can be a valuable predictor of the relatedness of species, whole genome sequences can be utilized to define and understand ancestral relationships between various organisms. Thus, efficient techniques that allow for automatic comparison and categorization of genes and species represented within extremely large sequence datasets must be explored.

Existing standard phylogenetic methods, such as maximum likelihood or Bayesian inference, which employ character-by-character comparisons, are computationally impractical for whole genomes, since most genomes contain millions and billions of sequence characters [SB03, SML02]. Most other methods that attempt to compare whole genomes

[1] http://www.ncbi.nih.gov/Genbank (National Center for Biotechnology Information)
[2] http://www.ebi.ac.uk/swissprot (European Bioinformatics Institute)

use “incomplete and arguably insufficient subsets of the available data” [SB03]. Moreover, many methods that construct phylogenies are based on alignments of similar sequences, and do not account for insertions or deletions of arbitrary size [SMB02].

A recently developed method based on the *Singular Value Decomposition* (SVD) [GL96], on the other hand, uses complete genome sequences, and generates gene and species phylogenies that are relatively unaffected by insertions or deletions [SML02, SMB02, SB03]. Using protein sequences, this method identifies significant independent characteristics from which evolutionary distances (between sequences) are computed.

More specifically, the method converts a large dataset comprising multiple genomes into a peptide-by-protein matrix A, where each protein is represented as a linear combination of short peptide string frequencies. By applying the SVD, matrix A is then factored into three components such that $A = U\Sigma V^T$. Matrices U and V provide new vector definitions for peptides and proteins, respectively. The orthonormal basis vectors of matrix U define independent characteristics or *correlated peptide motifs*[3] (copep motifs), and the orthonormal basis vectors of V describe clusters of related proteins or protein families. These clusters of related proteins share the correlated peptide motifs described by the corresponding vectors of U. By summing all the protein vectors corresponding to each species, species vectors can be obtained. The angles between these species vectors then give an estimate of species similarities [SB03].

As an alternative to the SVD, this study examines the use of the *Semi-Pivoted QR* (SPQR) decomposition deriving new vector definitions of peptides and proteins. The pep-

[3] A short conserved region in a protein sequence that usually correlates with a particular biological function or phenotype.

tide matrix factor is obtained by the application of SPQR algorithm to the original peptide-by-protein matrix A. A second application of the algorithm to A^T yields the protein matrix, which is ultimately used to construct both gene and species phylogenies.

Chapter 2

Motivation

An exhaustive similarity analysis can be used to define and understand the ancestral relationships of diverse organisms. In order to effectively compare and categorize genes and species, a balanced global analysis of significant independent characteristics within the protein sequences of species is required [SB03]. The SVD can be used to derive such an independent set of characteristics by factoring the input matrix (representing the entire dataset) as described in the previous chapter.

The SVD is known to produce an optimal factorization for *best-fit* reduced rank approximations to the original matrix [GL96]. However, it can be expensive to compute [BPS04]. A more efficient alternative to the SVD is a pivoted QR decomposition, which decomposes the input matrix into Q and R factors. By applying the algorithm in two ways, comparable results to the SVD can be obtained. The QR decomposition is particularly appealing as it conserves storage by not explicitly storing the dense factor Q, and recovering its actions later. Moreover, once an approximation of order k is computed, one automatically obtains

all the approximations of order $l < k$, whereas there is no easy way of increasing the size of approximations step-by-step with the SVD.

Timing examples on random sparse matrices [BPS04] indicate that the QR decomposition significantly outperforms the SVD. Indeed, the performance and memory analyses conducted in this study support that finding. The QR decomposition presents a scalable alternative to the SVD, and can be quite effective in comparing whole genomes in a phylogenetic context.

Chapter 3

Algorithm

The QR decomposition is a factorization that expresses the matrix A as $A = QR$, where Q matrix is orthonormal and R matrix is upper triangular. The classical methods used to perform QR decomposition include Householder transformations and the Gram-Schmidt algorithm. This chapter gives an overview of a *truncated pivoted QR decomposition* and the *Semi-Pivoted QR* algorithm that performs this decomposition.

3.1 Truncated Pivoted QR Decomposition

Given an $m \times n$ ($m \geq n$) matrix A, its pivoted QR decomposition is of the form [BPS04]

$$AP = QR,$$

where $m \times n$ Q is orthonormal (i.e., $Q^TQ = I$), R is an $n \times n$ upper triangular matrix, and P is a permutation matrix that reorders the columns of A (thus determines pivoting). A low

rank approximation to the matrix A can be achieved by partitioning the above factorization. Let $B = AP$, then B can be written as (see Figure 3.1) [Ste98]

$$(B_1^{(k)} B_2^{(k)}) = (Q_1^{(k)} Q_2^{(k)}) \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & R_{22}^{(k)} \end{pmatrix}. \tag{3.1}$$

The rank-k *truncated pivoted QR approximation* to B is then given by

$$\hat{B}^{(k)} = Q_1^{(k)} (R_{11}^{(k)} \;\; R_{12}^{(k)}). \tag{3.2}$$

The objective of this study is to minimize the difference $B - \hat{B}^{(k)}$ in order to obtain the closest rank-k approximation. Since $Q_2^{(k)}$ is orthonormal, the Frobenius norm of the difference reduces to

$$\|B - \hat{B}^{(k)}\| = \|Q_2^{(k)} (0 \;\; R_{22}^{(k)})\| = \|R_{22}^{(k)}\|.$$

The approximation (3.2) can be computed using a *Quasi Gram-Schmidt* algorithm proposed by Stewart [Ste98, BPS04], which is a variant of the classical Gram-Schmidt algo-

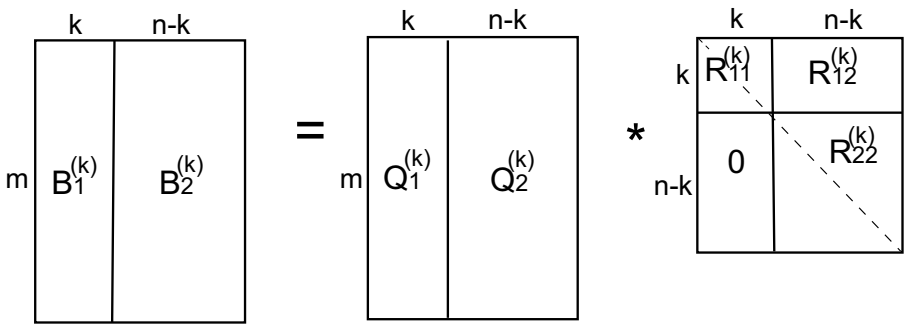

Figure 3.1: QR Decomposition

rithm. At each iteration, a column of A is used to generate an additional row of R and a column of Q, thus yielding the approximation in (3.2) at the end of iteration k.

The column of A at each step is chosen so that $\|R_{22}^{(k)}\|$ becomes small, that is, the column corresponding to the largest norm of $R_{22}^{(k-1)}$ is selected. This process of selecting columns of A, i.e., *column pivoting*, ensures that the strongly independent columns of A occur at the beginning of AP. Fortunately $R_{22}^{(k-1)}$ does not have to be computed explicitly due to the relation $\|B_j\| = \|R_j\|$, where B_j is the j^{th} column of B and R_j is the j^{th} column of R. However, using this strategy, if the norms of the columns of matrix A are approximately equal, the reduced rank approximation cannot be more accurate than 10^{-8} [BPS04].

Instead of calculating the complete approximation to A, suppose only $Q_1^{(k)}$ is needed to compute projections. Since

$$Q_1^{(k)} = B_1^{(k)} R_{11}^{(k)-1}, \tag{3.3}$$

only $R_{11}^{(k)}$ needs to be stored so that the multiplications by $Q_1^{(k)}$ can be recovered implicitly. Thus, the quasi Gram-Schmidt method is tailored to compute the R factor without forming the Q matrix. The algorithm is particularly appealing as it only involves sparse matrix-vector products (since dense products involving Q_1 are replaced by sparse products involving columns of B_1) and triangular system solutions, and leaves the original matrix A unchanged. The algorithm is discussed in more detail in the following section.

3.2 Semi-Pivoted QR Algorithm

The *Semi-Pivoted QR* (SPQR) algorithm used in this study (adapted from [Ste98]) is presented in Appendix A. It takes the original matrix A, the number of iterations to be performed and the error tolerance value as input, and returns an R_{11} factor, the permutation matrix P corresponding to column interchanges of A, and the norm of R_{22} (serving as the approximation error) as output.

In the first iteration, the pivot column a is determined and $R_{11}^{(1)}$ is set to $\|a\|$ and $Q_1^{(1)}$ is set to $a/\|a\|$. For subsequent iterations, the algorithm performs the following steps:

1. Select the column of A to be appended and pivot.
2. Perform quasi-Gram-Schmidt step.
3. Update R_{11}.
4. Compute the k^{th} column of Q_1.
5. Compute the k^{th} row of R_{12}.
6. Downdate the column norms of R_{22}.
7. Check for termination.

The quasi-Gram-Schmidt method is based on the classical Gram-Schmidt algorithm, however it has several differences. First, since the Q_1 factor is not formed explicitly, the method simulates its actions in the classical Gram-Schmidt via sparse matrix-vector multiplications and solutions of triangular systems of order k. Second, as a result of a cancellation in statement 16 (see Appendix A), the calculated q is not guaranteed to be orthogonal

to the rest of the columns of Q_1. Therefore, the Gram-Schmidt process is repeated on q in attempt to *reorthogonalize* it (statements $17 - 20$).

The algorithm then proceeds with updating the R_{11} factor (statements $22 - 23$) and computing the next column of Q_1 (statements $25 - 26$). It should be noted that even if the R_{12} matrix is not needed, it still has to be computed (statement 29) in order to compute the norm of R_{22} (approximation error). The column norms of R_{22} are downdated by subtracting the squares of the kth row of R_{12} from the squares of the corresponding R_{22} column norms and taking the square root (statements 31 and 33). The negative downdates are set to zero in order to prevent the selection of the corresponding columns of A in the next iterations.

At the end of each iteration the Frobenius norm of R_{22} is calculated. If it is less than the tolerance value specified by the user, the algorithm terminates, thus producing the R_{11} factor that forms a sufficiently accurate approximation to the original matrix A. However, if the algorithm runs for the number of iterations specified by the user without reaching the stopping criterion, the error in the approximation may certainly be greater than the requested tolerance. In this case, the norm of R_{22} can be examined to assess the error in the approximation.

Thus the SPQR algorithm only returns the R factor of the decomposition and in order to conserve storage, the Q factor is not stored explicitly. This is because even if the original matrix A is sparse, Q is likely to be dense. Instead, the relation in Equation (3.3) can be used to recover the matrix Q.

By applying it twice, the SPQR algorithm can be used to approximate the left and right singular vectors of the matrix A. Let the rank-k approximation to A be $A_k = U_k \Sigma_k V_k^T$.

Suppose the SPQR algorithm is applied to A to obtain a matrix X from the columns of A, and an upper triangular matrix R corresponding to R_{11}, where X and R are of order k (i.e., have k columns). Similarly, suppose the same algorithm is applied to A^T to get a matrix Y^T from the rows of A, and another upper triangular matrix S, where Y^T and S are of order k. Let $P \equiv XR^{-1}$ and $Q \equiv YS^{-1}$. Then, the lower rank approximation of A can be written as PWQ^T, where $W = R^{-T}X^TAYS^{-1}$ [BPS04]. By computing the SVD of $W = M\Sigma N^T$, $K \equiv PM$ can be used as a close approximation to U_k, and $L \equiv QN$ as a close approximation to V_k [BPS04]. However, in this study P and Q are used for approximating U_k and V_k, respectively.

3.3 Implementation Issues

The original Matlab implementation of the algorithm described above originated with Stewart in [BPS04]. However, for the purposes of this study, it has been converted to C++ in order to achieve more efficiency and scalability. Sparse peptide-by-protein matrices for input are provided in the Harwell-Boeing[1] format, where the matrix is represented in a compressed column storage format. Hence, the C++ code is implemented to account for sparse matrix-vector operations. More specifically, methods were created to calculate matrix-vector products of the form $A[:, P[start : end]] * x$ and $x^T * A[:, P[start : end]]$, where A is a sparse matrix, P is a permutation matrix, x is a dense vector, and $start$ and end parameters refer to starting and ending column indices, respectively.

[1]Harwell-Boeing format is a popular data structure used to store sparse matrices. Refer to [Mar04] for format specifications.

For solving triangular linear systems of equations in the SPQR algorithm, the Linear Algebra PACKage (LAPACK) [ABB+00] is used. The Innovative Computing Laboratory at the University of Tennessee developed and maintains LAPACK by supplying routines for solving systems of various types of linear equations. The DTPTRS module of LAPACK is called by the C++ program to solve a triangular system, where the triangular marix is stored in *packed* format.

Chapter 4

Whole Genome Phylogeny

With the production of increasing amounts of complete genome sequence data, there is an expanding need for comparing and categorizing genes and species in order to study the interrelationships among biological entities. *Phylogenies*, also known as evolutionary trees, play a major role in representing the evolution of a group of species from a common ancestor. *Phylogenetic trees* are branched dendrograms (treelike diagrams) that illustrate patterns of relatedness, where all organisms are placed at the leaves of the tree and each split of a branch is binary. Branch lengths of the phylogenetic trees are proportional to the predicted evolutionary time between organisms. Phylogenetic trees can be *rooted*, meaning that a unique organism corresponds to the most recent common ancestor of all the other organisms in the tree, or *unrooted*, meaning the common ancestor is unknown. Figure 4.1 depicts a simple tree for the three species Mouse, Rat, and Human. In this study, branch lengths of the tree are ignored, thereby placing more emphasis on the branching pattern of the tree (i.e., topology) rather than the evolutionary time.

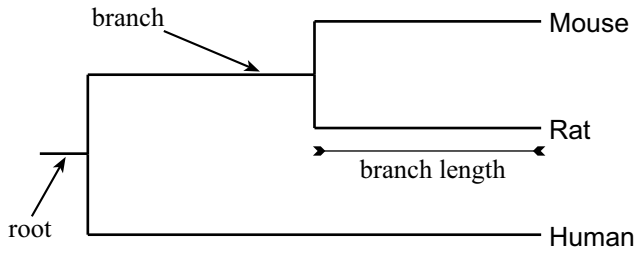

Figure 4.1: Sample Phylogenetic Tree

Besides being fundamental to the understanding of evolution and helping researchers organize their knowledge of diverse organisms and genomes, phylogenetic trees empower biologists to make important discoveries. They are one of the crucial components in pharmaceutical research and medicine with applications ranging from drug discovery and influenza vaccine development [Hil99] to tracking the origin and spread of emerging diseases and genetic defects. Phylogenetic trees also shed light in prediction of functions of novel genes by examining their similarity to known genes. Last but not least, phylogenies are useful in designing enhanced organisms and prove valuable in forensics and linguistics.

Constructing phylogenetic trees using genomic sequence data of species is based on the theory that sequence similarity can be a predictor of the relatedness of species. Currently, the character sequences most widely used for phylogenetic analysis arise from proteins and DNA. In this study, species trees are built using amino acid sequences in proteins, determined by the genetic code of the species in question. Instead of considering each amino acid in a protein sequence individually, all possible combinations of short contiguous strings of amino acids, n-gram peptides, are used, where n is the number of amino acids in

the string. For instance, since there are 20 common amino acid molecules, there are total of 20^4 = 160,000 overlapping tetrapeptides (4-gram peptides). Similarly, if 5-gram peptides are considered, there are 20^5 = 3,200,000 possible pentapeptides. Each protein can then be represented as a vector of peptide frequencies. That is, with a tetrapeptide encoding, a protein i can be written as $p_i = [f_{1i}, f_{2i}, ...f_{160,000i}]$, where f_{ki} is a frequency of occurrence of tetrapeptide k in protein i. The n proteins encoded by m peptides in a dataset then can form the columns of an $m \times n$ peptide-by-protein matrix A, which is typically very large and sparse. Hence, the entire dataset can be represented as a matrix in which each protein is uniquely represented as a vector of overlapping n-peptide frequencies.

As described before, genes and species can be effectively compared and categorized using significant independent characteristics within the protein sequences. In order to define peptides and proteins by these independent characteristics, a reduced rank approximation to the original matrix A can be obtained. This can be achieved by factoring the matrix using the SPQR algorithm described in Section 3.2. As before, let $P = XR^{-1}$, where X corresponds to a representative set of k columns of A, and R is an upper triangular matrix returned by the algorithm. Then, matrix P can be interpreted as a "peptide" matrix as it provides new vector definitions for n-gram peptides. Orthonormal basis vectors of P describe *correlated peptide motifs*, or *copep motifs* as particular linear combinations of all possible overlapping n-gram peptides [SB03, SMB02, SML02].

Furthermore, by the second application of the SPQR method to A^T, $Q = YS^{-1}$ is obtained, where Y is a representative set of k columns of A^T and S is an upper triangular matrix. Then, Q can be interpreted as a "protein" matrix, which serves to define a reduced

protein space, in which all the proteins of the dataset are accurately described as multi-dimensional vectors. By summing all the protein vectors corresponding to each species, representative species vectors are obtained. The cosine values between the angles of these species vectors then give an estimate of the pairwise similarity of species, which is used to construct a phylogenetic tree (explained in more detail in the following section).

The need for A^T for building phylogenetic trees, however, is problematic. The SPQR algorithm accesses a column of a matrix at each iteration, hence it needs a row of A when SPQR of A^T is being calculated. Since A is stored in compressed column format, retrieving a row of A no longer makes the access of values within the matrix contiguous. Several solutions to the problem are addressed in [BPS04], options including transposing the matrix in place and writing a row-oriented version of SPQR. When the matrix is transposed in place $O(nnz)$ additional work is required, where nnz is the number of non-zeros in the matrix [BPS04]. Returning to the whole genome phylogeny application, if both phylogenetic tree and copep motif analysis is required, one of those options has to be employed. If one only needs to construct trees, however, then the transpose of the matrix can be directly constructed from the data and factored. Copep motif analysis is not the main focus of this work and therefore is not discussed further. Nevertheless, the SPQR-based decomposition of A is run for the purposes of analyzing performance and memory usage to compare against the SVD method. For constructing phylogenetic trees, the transposes of all the matrices are calculated and stored in column compressed format. Thus, the existence of A^T is assumed.

As discussed above, the SPQR algorithm avoids storing the Q factor, and instead stores the much smaller factor R. But in this work the matrix Q is written to disk instead, since the

tree building program expects a binary file containing the matrix (using the above notation for SPQR of A^T). It is important to note that this feature of the algorithm is still useful as the Q factor is not accumulated in memory. However, if the disk storage is an issue, one can store the S matrix instead, and modify the tree building programs to compute the columns of Q using a triangular solve involving the columns of Y and a triangular matrix S. This would conserve space at the expense of time spent performing extra computation to obtain each column of Q.

4.1 Phylogenetic Tree Construction Process

The following steps illustrate the procedure of constructing phylogenies starting with the raw protein sequence data to the final product trees. A brief description of all the software used is presented in Table 4.1.

Table 4.1: Programs Used for Constructing Phylogenetic Trees

Program Name	Brief Description [Language or Package]
AACODE3	Parses a data file and creates a Harwell-Boeing file [C] with peptide-by-protein frequency matrix.
COSDIST	Computes pairwise cosine values between protein or species vectors and constructs an evolutionary pairwise distance matrix. [C]
NEIGHBOR	Generates a phylogenetic tree from a distance matrix. [PHYLIP]
CONSENSUS	Produces a consensus tree from multiple input trees. [PHYLIP]
TREEDIST	Computes distances between trees. [PHYLIP]
TREE EXPLORER	Displays, plots, rearranges and edits trees.

Step 1. Whole genome protein sequences for the species in question are downloaded from the National Center for Biotechnology Information (NCBI) and the text file is created in a predefined format retaining GeneBank ID numbers and species names. Using the program AACODE3, the text file is parsed and the dataset is represented using a large sparse peptide-by-protein matrix. In addition to specifying the peptide size, AACODE3 provides an option to apply log-entropy weighting [BB99] to the elements of a matrix and to normalize its columns. The resulting matrix A is written to a file in Harwell-Boeing format (compressed column storage representation).

Step 2. The SPQR algorithm is applied to A^T. At each iteration, a calculated column of Q is appended to a binary file. For k dimensions (iterations), the storage required for Q is $n * k * sizeof(double)$ bytes, where n is the number of columns in A (i.e., number of proteins). When the algorithm terminates, Q represents a "protein" matrix in which each protein (column of Q) is described as a particular linear combination of correlated peptide motifs.

Step 3. Protein vectors at a given dimension are summed for each species using the COSDIST program. At this point each species in the dataset is represented by a species vector.

Step 4. Pairwise cosine values are computed by the same COSDIST program for each pair of species vectors derived in Step 3. The larger the cosine value between two vectors, the smaller the angle between them, and hence they are considered more similar. These cosine values are subsequently used to build evolutionary pairwise distance matrices for each dimension using the conversion: $d_{ij} = -ln[(1 + cos\ \theta_{ij})/2]$ [SB03, SMB02, SML02].

The distance values are between 0 and 1 with $d_{ij} = 0$ indicating that proteins i and j are identical.

Step 5. Using the NEIGHBOR program from the PHYLIP software suite, species trees are generated from corresponding distance matrices. The Neighbor Joining (NJ) option of this program is used, which constructs a tree by successive clustering of lineages [Fel04]. It produces an unrooted tree and is very efficient as it "requires a time only proportional to the square of the number of species."

Equally important, before building the species trees, one may find it helpful to construct gene trees. Examining the protein families emerged in the gene trees may aid in determining the optimal number of dimensions for species tree construction. Gene trees are built using pairwise distance matrices obtained using all of the protein vectors in Q. In this case, the Unweighted Pair Group Method using Arithmetic Mean (UPGMA) option of the NEIGHBOR program is used in which a tree is generated by "successive (agglomerative) clustering using an average-linkage method of clustering" [Fel04]. Since prior expert knowledge is needed concerning the protein family memberships that are believed to be true for a particular dataset, gene trees are not generated in this particular study.

Step 6. The result from the previous species tree construction step is not a single optimal tree, but rather a set of species trees each corresponding to different dimensions (columns of Q). Any of the species trees generated could represent an equally believable estimate of the true tree. Considering a single tree provides no information about the support of individual clusters and as previous analyses indicate, accurate trees usually require hundreds of dimensions [SB03]. In order to identify a robust result, the approach usually

taken is to represent the set of trees by a single consensus tree. This is done using a CONSENSUS program found within the PHYLIP suite. The Extended Majority Rule option is selected, which produces a consensus tree consisting of all the groups that occur more the 50 percent of the time [Fel04]. Nonetheless, groups occurring less than 50 percent of the time are also included "as long as they continue to resolve the tree and do not contradict more frequent groups." [Fel04]

Step 7. For analyzing the effect of dimensions used to create phylogenetic trees, the TREEDIST program (also from the PHYLIP suite) was used to compare individual trees (from Step 5) within each other and with the consensus tree (from Step 6). Distances between trees were computed using the Symmetric Difference option of the program, which uses tree topologies and counts the number of differences within all possible branches that occur in the two trees [Fel04].

Step 8. Finally, the resulting trees are visualized and edited using TREE EXPLORER software [Tam99].

4.2 Example

This section demonstrates the phylogenetic tree construction process with the use of a small example based on seven land plant species. The corresponding dataset comprises a subset of the chloroplast protein coding content from a defined set of fast evolving genes. This dataset is considered in [GHBS04] using an SVD-based encoding method. There are a total of 28 protein sequences from the following species: *Chlorella vulgaris (Cvul), Cyanophora paradoxa (Cpar), Marchantia polymorpha (Mpol), Mesostigma viride (Mvir),*

Nephroselmis olivacea (Noli), Nicotiana tabacum (Ntab), and Pinus thunbergii (Pthu). For this problem 4-gram peptides are used yielding 160,000 possible overlapping tetrapeptides.

First, the AACODE3 program is used to parse the protein sequences and to generate a 160,000 $\times$ 28 log-entropy transformed matrix. The transpose of the matrix is then calculated and stored to be used by the SPQR algorithm. For the sake of demonstrating the algorithm easily, SPQR is run for only nine iterations giving the 28×9 protein matrix. Taking seven leading dimensions from resulting protein vectors, species vectors are obtained and the following pairwise evolutionary distance matrix is computed via COSDIST program:

	$Cvul$	$Mpol$	$Mvir$	$Noli$	$Ntab$	$Pthu$	$Cpar$
$Cvul$	0	0.306647	0.693147	0.800419	0.693147	0.693147	0.773147
$Mpol$	0.306647	0	0.310735	0.608681	0.693147	0.693147	0.628656
$Mvir$	0.693147	0.310735	0	0.693147	0.693147	0.693147	0.693147
$Noli$	0.800419	0.608681	0.693147	0	0.305800	0.305800	0.361688
$Ntab$	0.693147	0.693147	0.693147	0.305800	0	0.000000	0.693147
$Pthu$	0.693147	0.693147	0.693147	0.305800	0.000000	0	0.693147
$Cpar$	0.773147	0.628656	0.693147	0.361688	0.693147	0.693147	0

The pairwise distances are used as inputs to the NEIGHBOR program to produce the phylogenetic tree shown in Figure 4.2(a). Following [GHBS04], the tree is rooted with $Cpar$. The branch between $Ntab$ and $Pthu$ is evident from the distance matrix above, where the distance between these species is 0. The next closest species to both of these

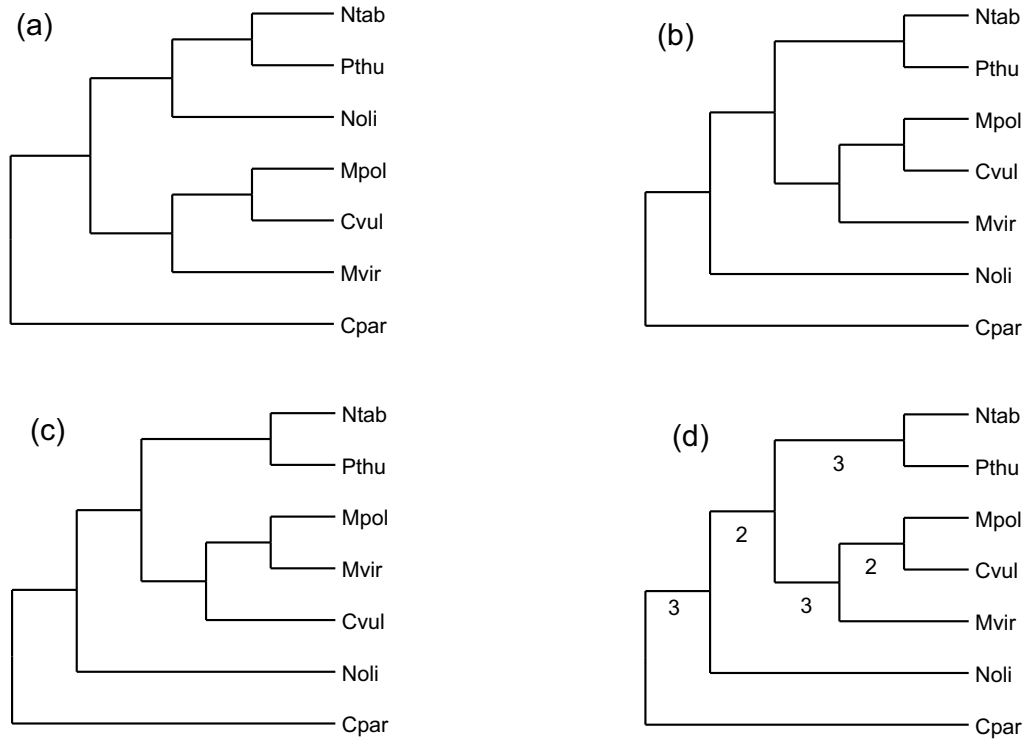

Figure 4.2: Phylogenetic Trees for the Plant Example. (a) Tree constructed using 7 dimensions, (b) Tree constructed using 8 dimensions, (c) Tree constructed using 9 dimensions, and (d) A consensus tree based on 7 to 9 dimensions.

species is $Noli$ (with distance 0.305800) and hence the corresponding branch in the tree. In a likewise fashion, the procedure is repeated for dimensions 8 and 9, and corresponding phylogenetic trees are illustrated in Figure 4.2(b) and (c). Since the trees slightly differ from each other, it might be helpful to construct a consensus tree. Figure 4.2(d) shows such a tree constructed from all three trees. In this case, the consensus tree is the same as the second tree. The numbers on the branches of the consensus tree indicate support values. For instance, all three trees have the branch between $Ntab$ and $Pthu$, but only two of them have the branch between $Mpol$ and $Cvul$. The symmetric distance between the first and second tree is two and the distance between the first and third tree is four. The reader should note that the phylogenetic trees in this example may not be the optimal ones, and one can possibly find better trees by running the SPQR algorithm for more iterations and/or using different dimension trees to derive a consensus tree.

Chapter 5

Experimental Results

Three experiments were conducted for datasets representing land plant whole genomes, eukaryote whole genomes, and vertebrate mitochondrial genome sequences. Whole genome phylogeny analyses for these problems had been previously conducted using an SVD-based method. The SPQR-based analysis considered here was designed to compare the two approaches in terms of the accuracy of the tree results, computational speed, and working memory usage.

SPQR, COSDIST, NEIGHBOR, CONSENSE, and TREEDIST programs (see Section 4.1) were run on the *boba* cluster of the Scalable Intracampus Research Grid (SInRG)[1] at the University of Tennessee, Knoxville. The boba cluster is comprised of 32 2.4 GHz dual Xeon Pentium IV processors with 2GB of RAM, 512KB of cache, and 2 73GB MAXTOR 6L080J4 disk drives supported by a 1Gbs interconnection network. Tree graphics using TREE EXPLORER (see Section 4.1) were produced on a desktop PC. The specifications

[1] http://www.cs.utk.edu/sinrg

for each problem considered, the tree results, as well as the performance and memory usage analyses conducted are presented in subsequent sections.

5.1 Problem Specifications

The problems were selected to study the effects of varying dataset types and sizes. The size of the dataset is determined by the number of proteins considered and the length of overlapping peptides used to encode these proteins. This section provides specifications for species and their protein content for each problem.

5.1.1 Land Plants

Among the biological community, the chloroplast genome of the plant genome has been vital in studying plant evolution and systematics. [GHBS04] conducts a whole genome phylogeny of land plants based on the SVD analysis of the entire protein content of chloroplast genomes. The dataset consists of 17 species, which amount to a total of 1,675 proteins. The species names, along with the number of proteins regarded for each of the species, is listed in Table 5.1 [GHBS04]. *Chaetospiridium globusom* is later used to root the tree. All possible combinations of overlapping tetrapeptides are considered in the study, thus yielding a $160,000 \times 1,675$ peptide-by-protein data matrix whose elements are log-entropy transformed (see Section 4.1).

Table 5.1: List of Species in Land Plant Dataset

Species	Abbreviation	No. of Proteins
Adiantum capilllus-veneris	Acap	87
Amborella trichopoda	Atri	86
Anthoceros formosae	Afor	90
Arabidopsis thaliana	Atha	87
Atropa belladonna	Abel	87
Calycanthus floridus	Cfer	88
Chaetospiridium globosum	Cglo	98
Lotus corniculatus	Lcor	82
Marchantia polymorpha	Mpol	89
Nicotiana tabacum	Ntab	101
Oenothera elata	Oela	117
Oryza sativa	Osat	108
Pinus thunbergii	Pthu	164
Psilotum nudum	Pnud	101
Spinacia oleracea	Sole	98
Triticum aestivum	Taes	83
Zea mays	Zmay	111

5.1.2 Vertebrates

An SVD-based method has been used to construct vertebrate phylogenies using 32 [SMB02] and 64 [SML02] mitochondrial vertebrate genomes. In this study, 64 whole vertebrate genomes consisting of 832 mitochondrial proteins are used. The dataset specifications are listed in Table 5.2. There are 13 protein sequences for each species. This time, each protein is represented as a linear combination of all possible overlapping pentapeptides producing the 3,200,000 $\times$ 832 peptide-by-protein matrix. Again, the log-entropy transformed version of the matrix is used.

Table 5.2: List of Species in Vertebrate Dataset

Species	Abbrev.	Species	Abbrev.
Alligator mississippiensis	Amis	Artibeus jamaicensis	Ajam
Aythya americana	Aame	Balaenoptera musculus	Bmus
Balaenoptera physalus	Bphy	Bos taurus	Btau
Canis familiaris	Cfam	Carassius auratus	Caur
Cavia porcellus	Cpor	Ceratotherium simum	Csim
Chelonia mydas	Cmyd	Chrysemys picta	Cpic
Ciconia boyciana	Cboy	Ciconia ciconia	Ccic
Corvus frugilegus	Cfru	Crossostoma lacustre	Clac
Cyprinus carpio	Ccar	Danio Rerio	Drer
Dasypus novemcinctus	Dnov	Didelphis virginiana	Dvir
Dinodon semicarinatus	Dsem	Equus asinus	Easi
Equus caballus	Ecab	Erinaceus europaeus	Eeur
Eumeces egregius	Eegr	Falco peregrinus	Fper
Felis catus	Fcat	Gadus morhua	Gmor
Gallus gallus	Ggal	Halichoerus grypus	Hgry
Hippopotamus amphibius	Hamp	Homo sapiens	Hsap
Latimeria chalumnae	Lcha	Loxodonta africana	Lafr
Macropus robustus	Mrob	Mus musculus	Mmus
Mustelus manazo	Mman	Myoxus glis	Mgli
Oncorhynchus mykiss	Omyk	Ornithorhynchus anatinus	Oana
Orycteropus afer	Oafer	Oryctolagus cuniculus	Ocun
Ovis aries	Oari	Paralichthys olivaceus	Poli
Pelomedusa subrufa	Psub	Phoca vitulina	Pvit
Polypterus ornatipinnis	Porn	Pongo pygmaeus abelii	Ppya
Protopterus dolloi	Pdol	Raja radiata	Rrad
Rattus norvegicus	Rnov	Rhea americana	Rame
Rhinoceros unicornis	Runi	Salmo salar	Ssal
Salvelinus alpinus	Salp	Salvelinus fontinalis	Sfon
Sciurus vulgaris	Svul	Scyliorhinus canicula	Scan
Smithornis sharpei	Ssha	Squalus acanthias	Saca
Struthio camelus	Scam	Sus scrofa	Sscr
Talpa europaea	Teur	Vidua chalybeata	Vcha

5.1.3 Eukaryotes

An SVD-based whole genome phylogeny method was applied in a previous study for categorizing nine complete eukaryotic nuclear genomes of diverse organisms ranging from human being to yeast [SB04]. The protein sequences used for this problem are all curated[2]. Table 5.3 adapted from [SB04] displays the species names and their protein content. Nine species composed of 175,559 proteins are encoded by a 160,000 $\times$ 175,559 peptide-by-protein matrix, which is log-entropy transformed and column-wise normalized.

5.2 Phylogenetic Trees

Initially, SPQR was run on each of the datasets for k iterations, where k is the number of singular triplets (i.e., basis vectors, motifs) retained by the SVD-based method for the corresponding dataset. However, for the SPQR-based approach, this number of iterations

Table 5.3: List of Species in Eukaryote Dataset

Species	Abbreviation	No. of Proteins
Homo sapiens (Human)	Hsap	25,319
Mus musculus (Mouse)	Mmus	25,371
Rattus norvegicus (Rat)	Rnov	21,204
Fugu rubripes (Pufferfish)	Frub	37,439
Anopheles gambiae (Mosquito)	Agam	16,091
Drosophila melanogaster (Fly)	Dmel	18,107
Caenorhabditis elegans (Worm)	Cele	21,124
Saccharomyces cerevisiae (Yeast)	Scer	5,855
Plasmodium falciparum (Malaria)	Pfal	5,049

[2]Curation refers to the process of verifying and enhancing data submitted to biological databases.

usually deemed not sufficient for constructing accurate phylogenies. The protein vectors with increased dimensions (i.e., more right singular vectors) were then created to obtain comparable trees to SVD consensus trees. Let a consensus tree based on x to y dimensions, where $x > y$, be called x-y consensus tree. To get an estimate of which subset of the trees to use for an SPQR consensus tree, the tree distances (see Section 4.1) were computed between the SVD consensus tree and each of the SPQR-derived trees corresponding to different dimensions.

For the land plants problem, the SVD method [GHBS04] yielded two sets of 534 basis vectors and a 10-534 consensus tree (shown in Figure 5.1). The distance between this tree and the 10-534 consensus tree produced from the SPQR-based method was 6. The distance value dropped to 2 when a 10-620 SPQR consensus tree was generated (see Figure 5.2), but identical trees could not be obtained even by increasing the number of iterations in the SPQR algorithm further. The only difference between the two trees is the lack of the branch between $Pnud$ and $Acap$ in the SPQR tree. However, it can be argued that the difference is not significant since this branch is not highly supported in the SVD tree.

A graph illustrating tree distances is displayed in Figure 5.3. For all of the trees derived from dimension settings between 10 and 620, distances between adjacent trees (a given tree and the next tree in the series) are plotted. Distances between each of these trees and the consensus tree in Figure 5.1, as well as the consensus tree in Figure 5.2 are also demonstrated. The trend lines were produced using a moving 20 point average. As the number of dimensions considered increases to 453, convergence between the corresponding SPQR-derived trees and the SVD-derived consensus tree can be observed. The lowest

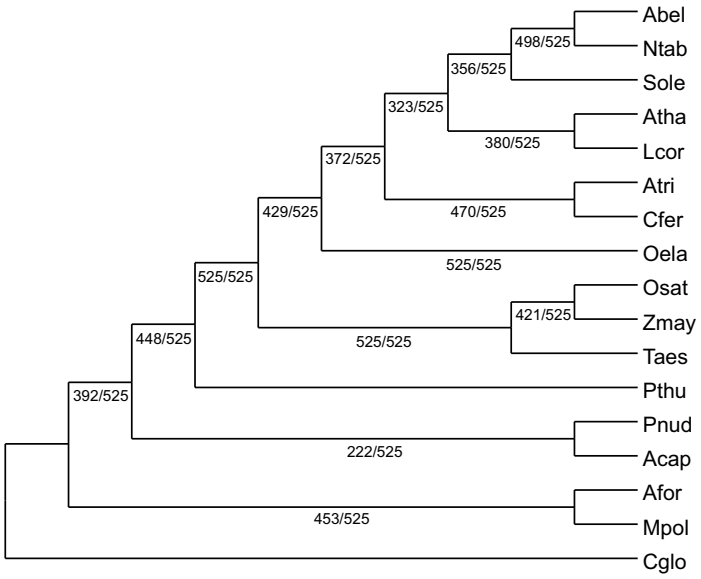

Figure 5.1: SVD Plant Tree

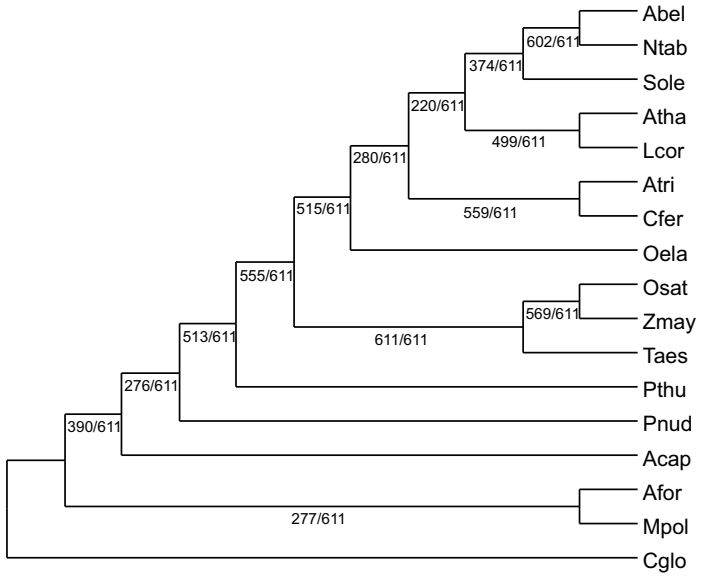

Figure 5.2: QR Plant Tree

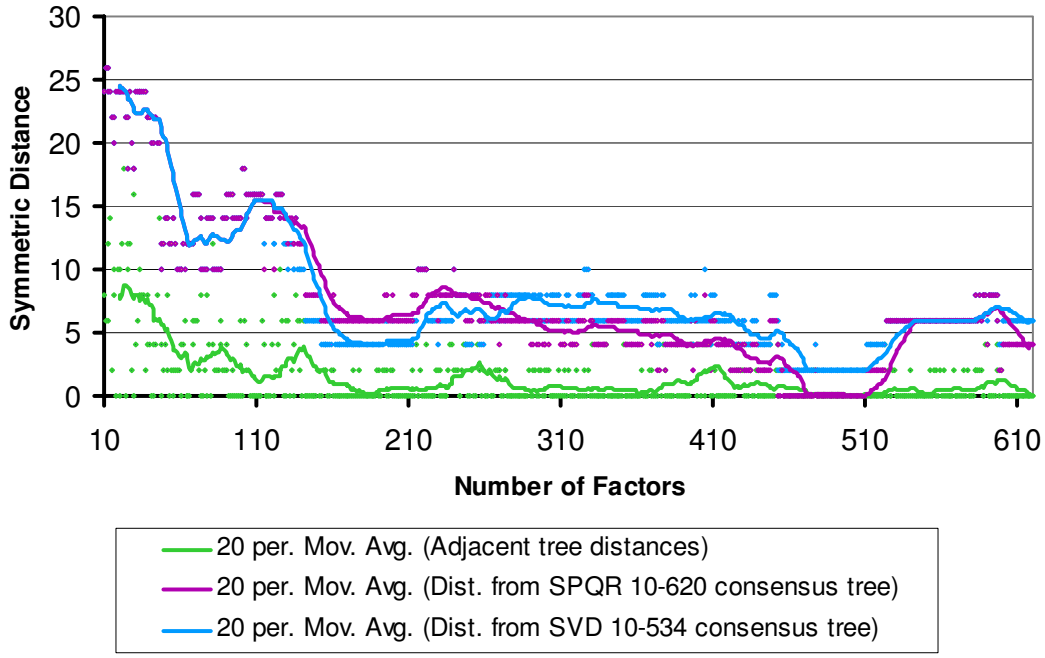

Figure 5.3: Plant Tree Distance

tree distance value of 2 is maintained between the SPQR-derived trees and the SVD-based consensus tree until 510 dimensions. However, as the dimensions are further increased, distances between the corresponding trees start to increase again.

For the vertebrate problem, a recent SVD-based analysis (unpublished) supplied 121 basis vectors and the resulting 10-121 consensus tree shown in Figure 5.4, where $Rrad$ is arbitrarily chosen as a root. Note that only the branches appearing in 50 percent or more of the trees are shown; the branches appearing in less than 50 percent of the trees (i.e., poorly supported branches) are collapsed into polytomies (nodes with more than two descendant branches). The 10-400 consensus tree derived from the SPQR method is shown in Figure 5.5. Although there are a few differences between the two trees, most of the species in

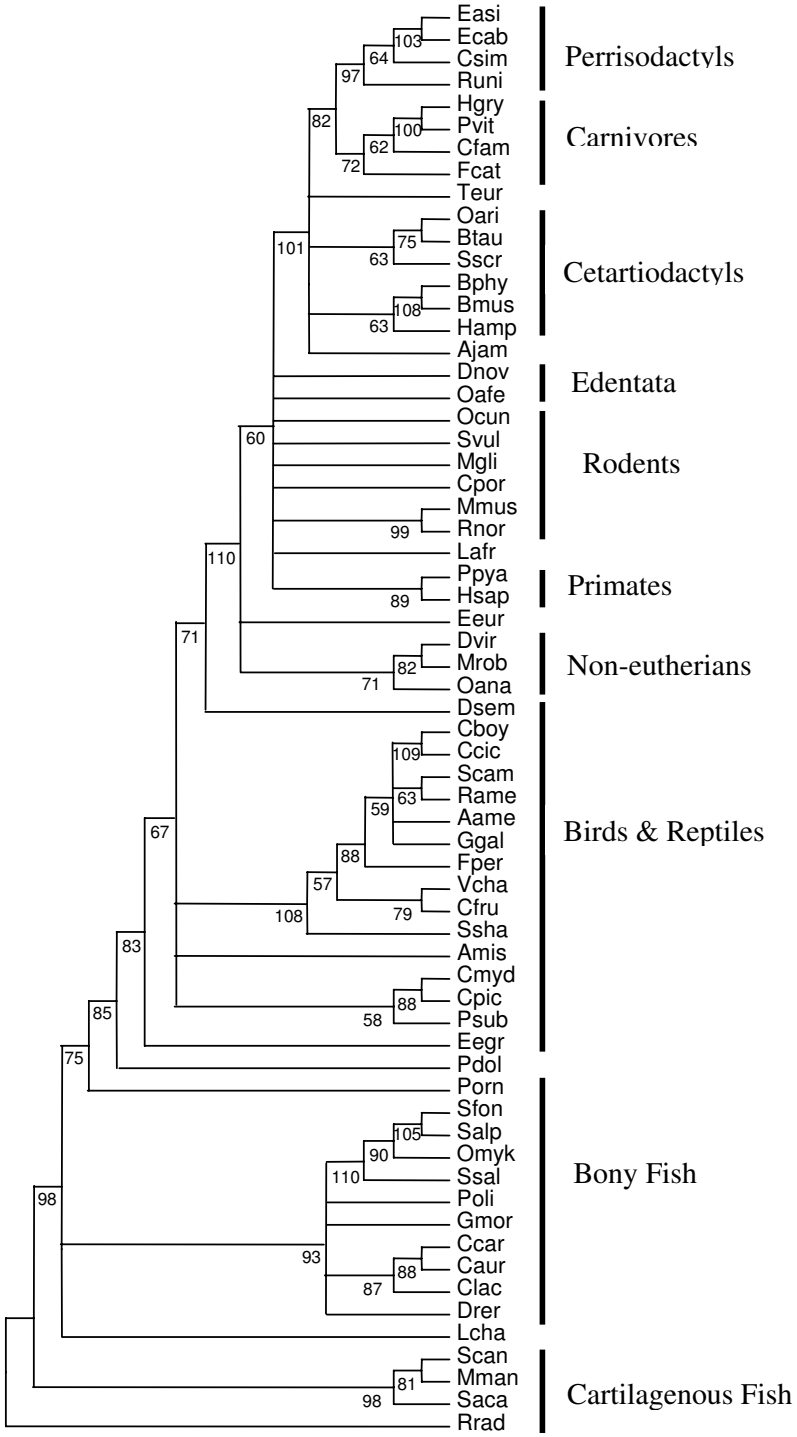

Figure 5.4: SVD Vertebrate Tree

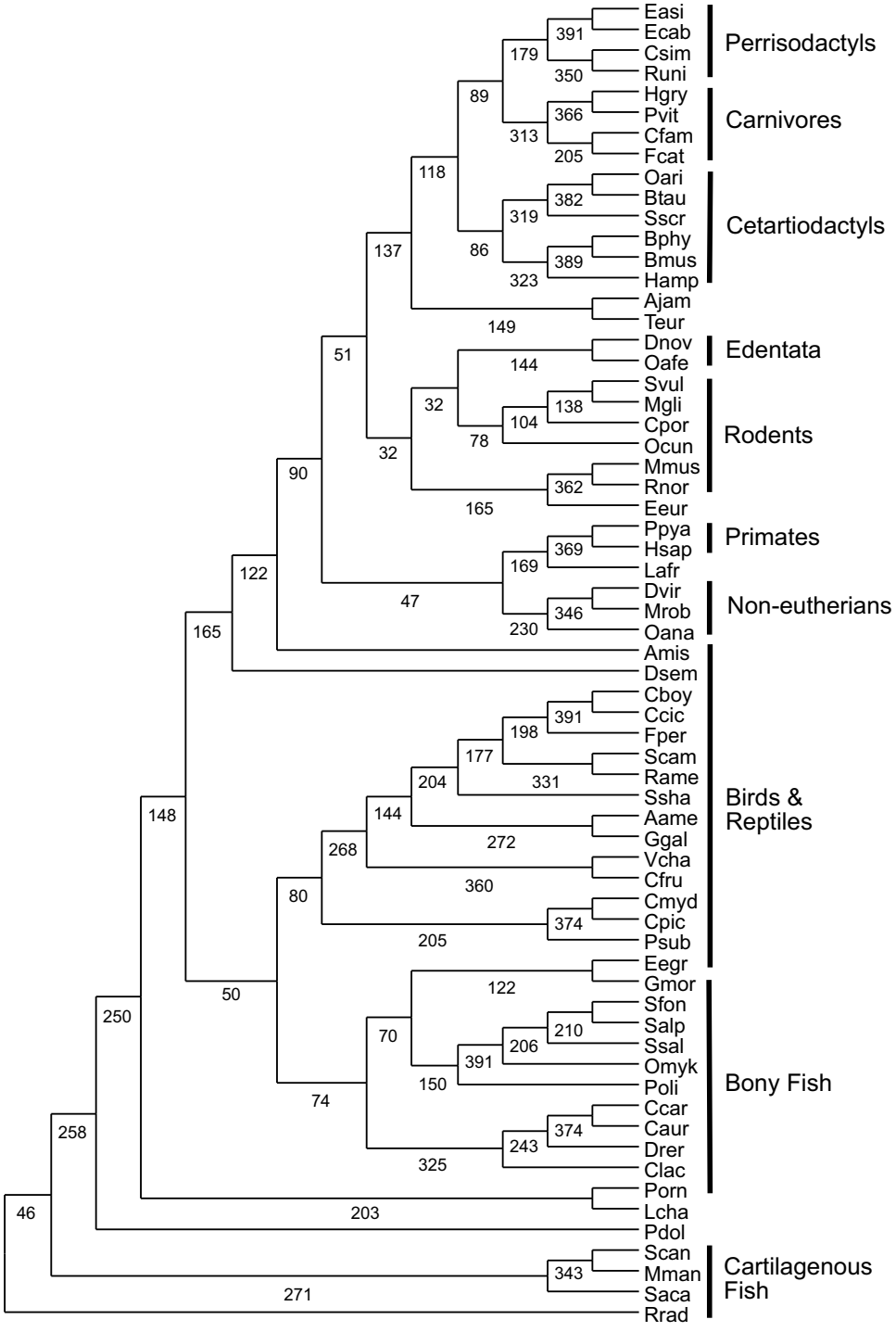

Figure 5.5: QR Vertebrate Tree

the SPQR tree are placed into the same well accepted groupings as the SVD tree. One of the problems in the SPQR tree is that the non-eutherian mammals like $Mrob$ (kangaroo) and $Oana$ (platypus) are not clustered exclusively at the base of the mammal portion of the tree like they do in the SVD tree. Instead, they appear at the base, forming a polyphyletic group (a group of organisms with different most recent ancestors) with other mammals. In addition, the $Eagr$ (lizard) groups with the bony fish instead of grouping with birds and reptiles, but this behavior is seen in other SVD-derived trees as well. A somewhat surprising tendency of $Lafr$ (elephant) to group with primates is also observed in the SVD-derived trees [SML02]. This is also eminent in the SPQR consensus tree.

The tree distances for the vertebrate problem (calculated as described above) are shown in Figure 5.6. One can see that as the number of factors increases, the distance from the SVD consensus tree slowly decreases. Yet the smallest distance value is still very large. This can be due to the fact that a non-collapsed version of the SVD-derived consensus tree is used to measure the tree distances (since the TREEDIST program does not accept polytomies as input). Consequently, the poorly supported (and possibly inaccurate) branches of the SVD-derived tree are equally treated as the well-supported branches when compared to the SPQR-derived trees. Then, different (but relatively unimportant) branching patterns within the identified basic groups may cause the SVD-derived consensus tree to appear "distant" from a particular SPQR-derived tree, even though it may be a reasonably good approximation of the "true tree." Therefore, it may be possible to find a better SPQR consensus tree by looking for the correct order and clustering of those basic groups instead of choosing the one that is closest to a particular SVD-derived tree.

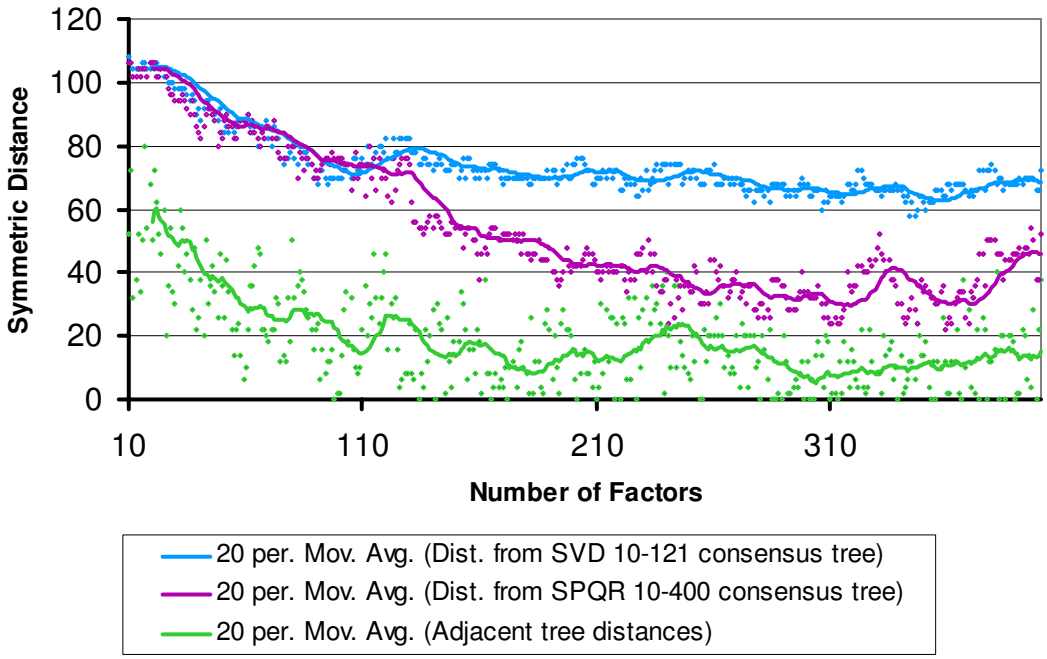

Figure 5.6: Vertebrate Tree Distance

Moreover, recall that optimal dimensions for creating consensus trees can be estimated by reference to prior categorical information concerning gene family memberships. Therefore, it might be possible to find a better consensus tree that surveys different dimensions supported by the gene tree analysis.

The SVD-based analysis of the dataset compiled using nine whole eukaryotic genomes produced 434 well-conserved motifs and gene families. The resulting 10-434 consensus tree [SB04] displayed in Figure 5.7, is equivalent to the universally accepted phylogeny of these nine species. The distance between this tree and a 10-434 SPQR consensus tree was 2, the only difference between the two trees being the swap of $Hsap$ and $Rnov$ in

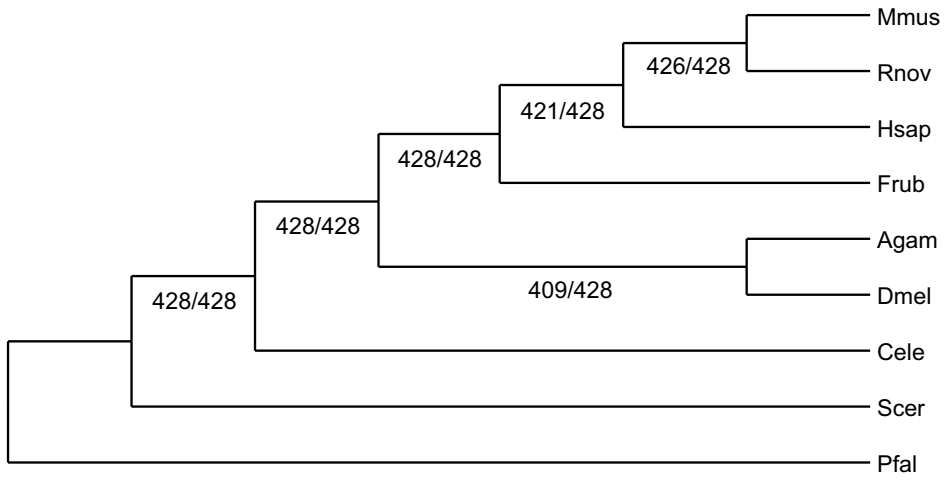

Figure 5.7: SVD Eukaryote Tree

the SPQR-derived tree. The SPQR algorithm was then run for further iterations, but there was no change in trees derived from dimension settings less than 1198. Starting from dimension 1198, identical trees to the SVD-derived consensus tree were produced. Figure 5.8 illustrates a 1198-1400 consensus tree derived from SPQR analysis and Figure 5.9 shows the corresponding tree distances (adjacent trees and from SVD-based consensus tree).

5.3 Performance Analysis

For each of the experiments described in the previous sections, the real time spent for the SPQR-based whole genome analysis was measured and compared against the corresponding SVD-based analyses. Specifically, the time taken to apply the SPQR algorithm on A and on A^T for the number of iterations specified in the previous section, time for running

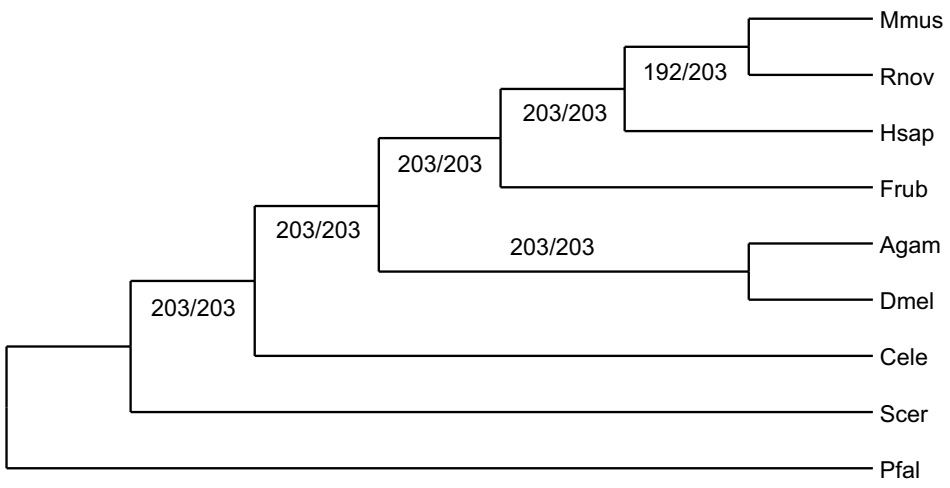

Figure 5.8: QR Eukaryote Tree

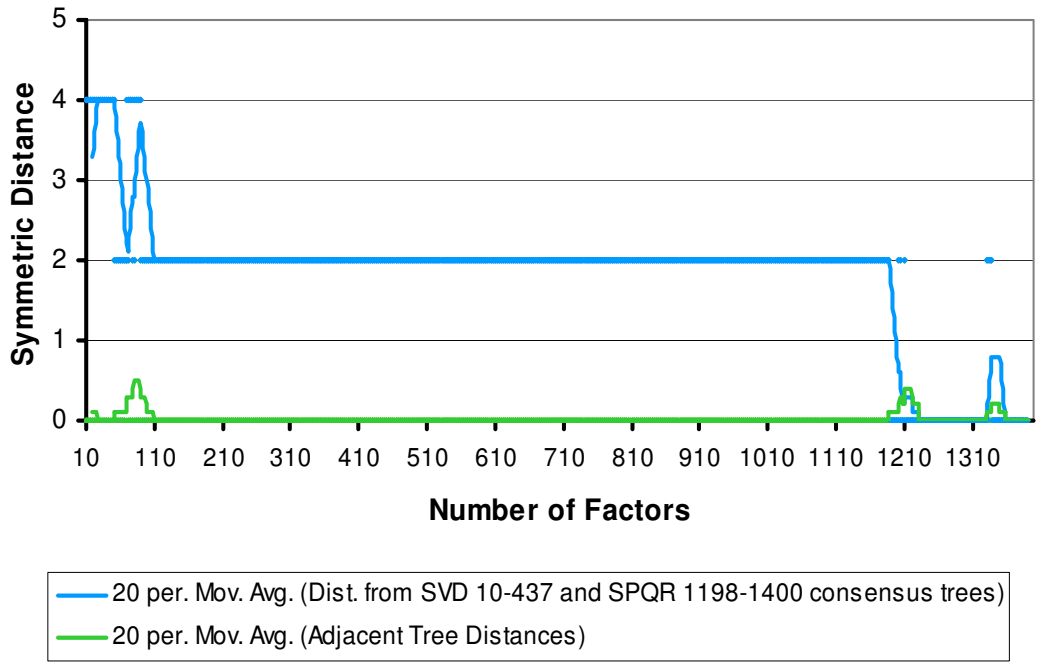

Figure 5.9: Eukaryote Tree Distance

the COSDIST program (for each dimension considered in the final consensus tree), and time taken to run the NEIGHBOR program (for each of the resulting trees) were recorded. The Linux *time* command was used to measure the elapsed real time. The tables with timings for each dataset are presented in Appendix B. Timings are also illustrated using bar charts for the land plant (Figure 5.10), vertebrate (Figure 5.11), and eukaryote (Figure 5.12) problems. The first bar (from the left) in the charts corresponds to the elapsed time for the SPQR algorithm on A^T, and time required by the COSDIST program for each dimension in question. The second bar corresponds to the time for the truncated SVD of A and the COSDIST program for each dimension (right singular vector) considered in the SVD-derived consensus tree. The NEIGHBOR program timings are not illustrated in the charts as they constitute only a very small portion of the entire runs. Recall that in this

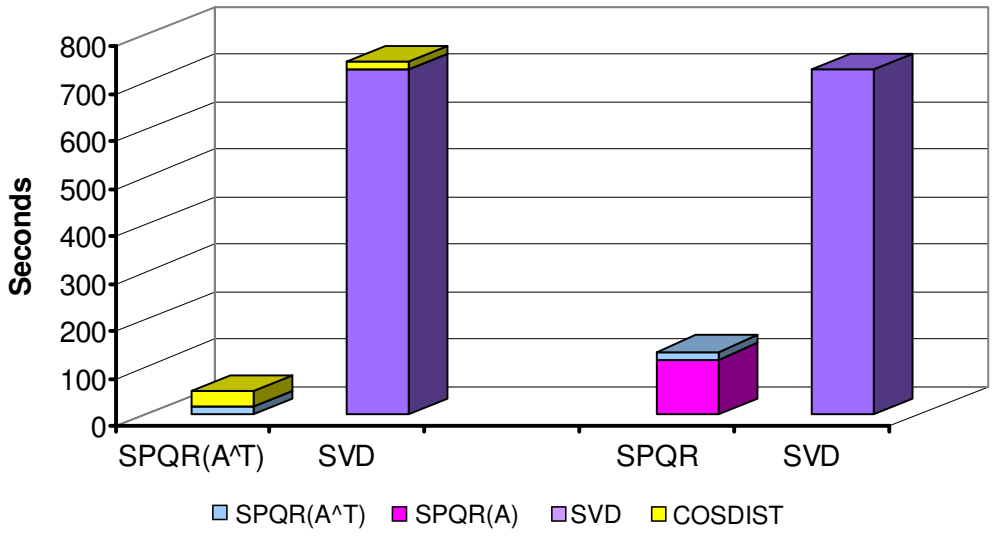

Figure 5.10: Plant Times

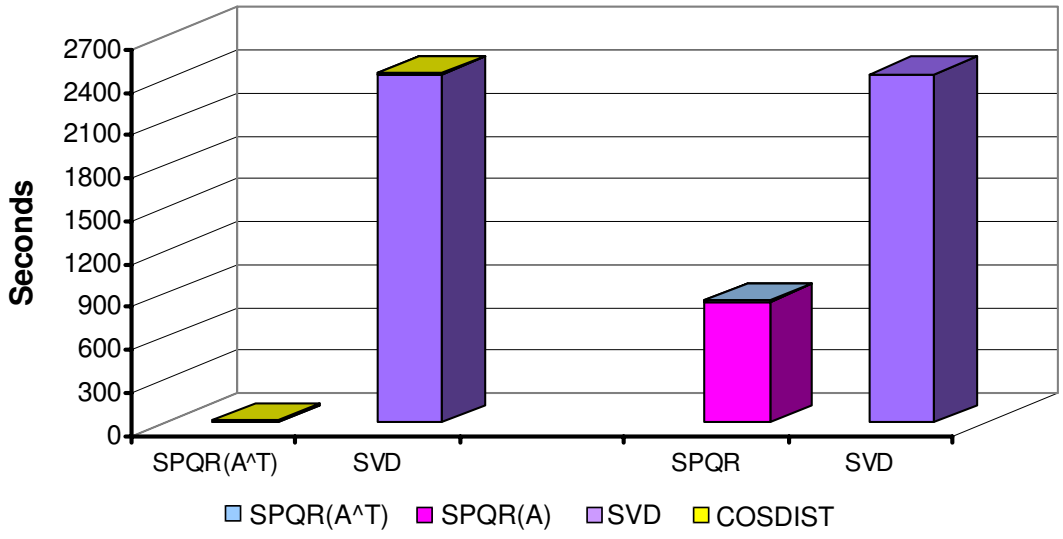

Figure 5.11: Vertebrate Times

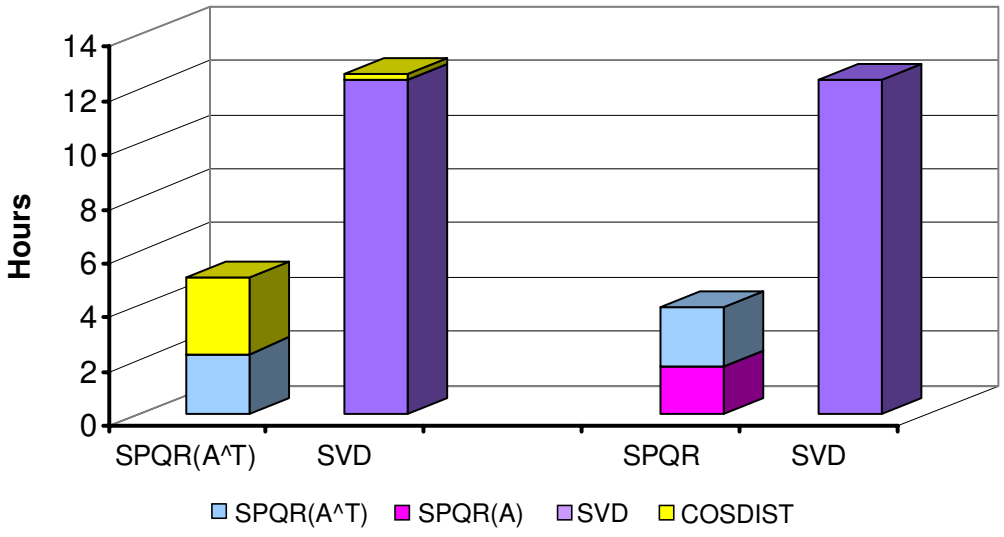

Figure 5.12: Eukaryote Times

particular study, it is assumed that transposes of the matrices are provided, thus the time taken to compute them is ignored. The third bar corresponds to the time required by the SPQR algorithm on A and A^T, and the fourth bar is the corresponding time required to compute the truncated (k dimensions only) SVD of A.

With respect to time, the SPQR-based whole genome phylogeny approach outperformed the SVD-based method for all three datasets. With the SPQR method, phylogenetic trees were constructed 13 times faster for plants, over 76 times faster for vertebrates, and almost 2.5 times faster for eukaryotes, even though protein vectors with more dimensions were created by the SPQR method (i.e., 620 vs. 534 for plants, 400 vs. 121 for vertebrates, and 1400 vs. 437 for eukaryotes). Notice, however, that the COSDIST program with the SPQR approach takes substantially longer, since cosine distances have to be computed using more dimensions. This is particularly evident in the eukaryote case, where the time taken to construct distance matrices using COSDIST program took 2.8 hours for the SPQR approach (which is longer than the time for running the algorithm itself), whereas it only took 10.5 minutes for the SVD approach. Thus, as the number of dimensions is increased, computing distance matrices becomes a bottleneck. Certainly, more storage is required to store these additional distance matrices as well. As the number of trees considered to build the consensus tree is increased, the NEIGHBOR program times increased proportionally.

On the other hand, if one needs both the protein and peptide matrices, the sum of the time taken for SPQR in factoring A and A^T can be compared against the time taken to compute the truncated SVD of A. In this case, SPQR outperformed the SVD by almost 5.5 times for plants, 118 times for vertebrates and over 3 times for eukaryotes. In addition, the

time taken for SPQR of A is longer than the time for SPQR of A^T for plants and vertebrates, while the opposite is true for eukaryotes. This can be explained by the dimensions of the matrices supplied for the algorithm. In the case of plants and vertebrates, the number of rows in the original matrix A is far greater than the number of columns (A is 160,000 $\times$ 1,675 for plants and 3,200,000 $\times$ 832 for vertebrates), and therefore, when SPQR of A is being computed, each column of the Q factor generated at each iteration has far more elements than the one computed by the SPQR of A^T. On the contrary, the number of columns in the eukaryote data matrix is greater than the number of rows (A is 160,000 $\times$ 175,559), and thus it takes longer to compute the protein matrix Q than the peptide matrix P (see Section 3.2).

5.4 Memory Usage Analysis

In addition to comparing the performance of the two methods, the working storage (RAM) required to run both the SPQR and SVD algorithms is measured in bytes. Figure 5.13 illustrates the memory usage of SPQR when it is applied to the original matrix A, its transpose, and the memory usage of SVD for each dataset considered. Memory occupied by the input matrices is excluded. For the SPQR algorithm, the memory requirements include the $k \times k$ triangular matrix R, the permutation vector with n elements, the vector of length m that contains a particular column of Q, and a few additional work vectors of length k and n, where k is the number of iterations supplied by the user, and m and n correspond to the number of rows and columns in the input matrix A.

One can see from Figure 5.13 that for the datasets considered, the SPQR algorithm

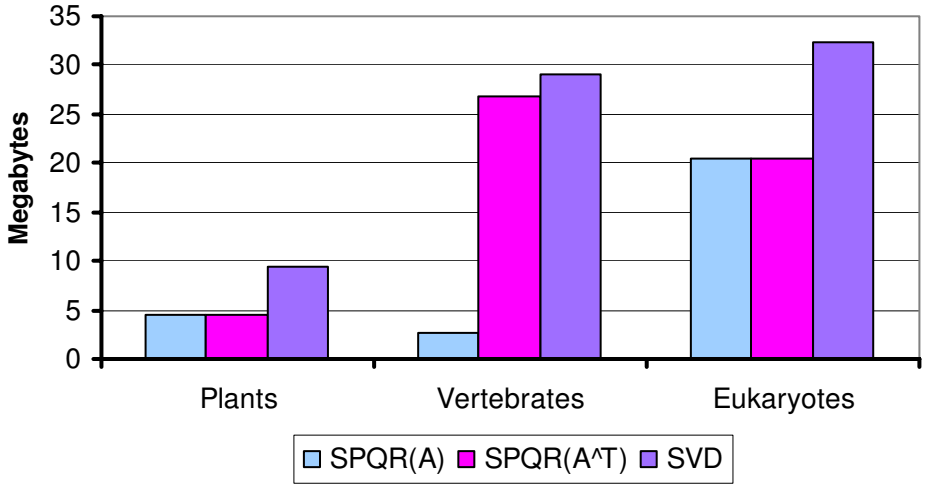

Figure 5.13: Memory Usage

requires less memory than the SVD algorithm. For plants, both SPQR of A and A^T use less than a half of the memory that the SVD algorithm uses. Concerning vertebrates, even though the SPQR of A does not save a good deal of working storage compared to the SVD, the SPQR of A^T, on the other hand, requires 11 times less memory than the SVD does. For eukaryotes, about 1.57 times less working memory is required for both SPQR of A and A^T. Again, one should keep in mind that the SPQR-based experiments generated significantly more basis vectors than did the SVD-based experiments. The differences would have been more drastic granted the same number of basis vectors had been generated by both algorithms.

As far as storage is concerned, it would be very easy to modify the current C++ program to store the R factor instead of storing the Q factor to disk. The required storage in that case would comprise only k^2 as opposed to $(n + m) * k$ floating point numbers. Then, the

COSDIST program would have to be modified to recover the Q using R, so that it could be used to compute the cosine distances (at a substantially higher cost in execution time).

Chapter 6

Conclusions

6.1 Summary

With whole genome sequence data rapidly accumulating in public databases, developing effective methods to categorize genes and species in extremely large datasets is a challenging problem. This study demonstrates how the truncated pivoted QR decomposition can be used as an alternative to the Singular Value Decomposition (SVD) to identify and define gene families as well as conserved motifs, and to generate gene and species phylogenies. The algorithm is implemented in C++, and is applied to land plant, vertebrate and eukaryote datasets composed of protein sequences. The phylogenetic trees, performance and memory usage results illustrated justify further development of the method and its application to other complex whole genome datasets.

6.2 Future Work

Currently, motif analysis has not been performed on the datasets considered. Yet the corresponding peptide matrices are readily available, and one can construct well conserved motifs for each dataset in the future. The result of the phylogenetic analysis described in this method is not a single optimal tree, but rather the set of all the "best" trees produced during the search. Ultimately, the phylogenetic analysis of all these trees becomes a bottleneck. With the use of prior gene family membership information, however, one can generate optimal or near-optimal phylogenetic trees. Likewise, by employing this method, one may be able to find "better" consensus trees for the datasets considered in this study.

One limitation resulting from the application of the algorithm to whole genome phylogeny and the use of sparse encoding is that the input matrix in Harwel-Boeing format must be transposed. Thus, another possible area for future research may involve examining the tradeoffs (between storage and additional computation) arising from this issue (see also [BPS04]), and employing the technique that is best suited for the particular whole genome phylogeny application.

Concerning its implementation, the algorithm could also be parallelized in order to achieve scalable performance for larger and larger problem sizes. Naturally, the algorithm could be applied to the matrix and its transpose in parallel, so that new vector definitions for peptides and proteins could be simultaneously generated.

Bibliography

Bibliography

[ABB+00] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Cruz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 2000.

[BB99] Michael W. Berry and Murray Browne. *Understanding Search Engines: Mathematical Modeling and Text Retrieval*. SIAM Book Series: Software, Environments, and Tools, first edition, 1999.

[BPS04] Michael W. Berry, Shakhina A. Pulatova, and G. W. Stewart. Computing Sparse Reduced-Rank Approximations to Sparse Matrices. Submitted to *ACM Transactions on Mathematical Software*, 2004.

[Fel04] Joe Felsenstein. Phylip. Available on: `http://evolution.genetics.washington.edu/phylip.html`, January 2004.

[GHBS04] Cynthia J. Gibas, Khildir W. Hilu, Michael W. Berry, and Gary W. Stuart. A Phylogeny of Land Plants Based on Whole-Genome Analysis of Chloroplast

Using Correlated Peptide Motifs. Virginia Polytechnic Institute and State University. In preparation, 2004.

[GL96] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. John Hopkins, Baltimore, MD, third edition, 1996.

[Hil99] David M. Hillis. Predictive Evolution. *Science*, 286(5446):1866–1867, December 1999.

[Mar04] Matrix Market. Harwell-boeing exchange format. Available on: `http://math.nist.gov/MatrixMarket/formats.html\#hb`, June 2004.

[SB03] Gary W. Stuart and Michael W. Berry. A Comprehensive Whole Genome Bacterial Phylogeny Using Correlated Peptide Motifs Defined in a High Dimensional Vector Space. *Journal of Bioinformatics and Computational Biology*, 1(3):1–19, 2003.

[SB04] Gary W. Stuart and Michael W. Berry. An SVD-based Comparison of Nine Whole Eukaryotic Genomes Supports a Coelomate rather than Ecdysozoan Lineage. Submitted to *Nature Genetics* as a Brief Communication, January 2004.

[SMB02] Gary W. Stuart, Karen Moffett, and Steve Baker. Integrated Gene and Species Phylogenies from Unaligned Whole Genome Protein Sequences. *Bioinformatics*, 18(1):100–108, 2002.

[SML02] Gary W. Stuart, Karen Moffett, and Jeffery J. Leader. A Comprehensive Vertebrate Phylogeny using Vector Representations of Protein Sequences from Whole Genomes. *Molecular Biology and Evolution*, 19:554–562, 2002.

[Ste98] G. W. Stewart. Four Algorithms for the Efficient Computation of Truncated Pivoted QR Approximations to a Sparse Matrix. Technical Report TR-98-12, Institute for Advanced Computer Studies, University of Maryland, College Park, MD, April 1998.

[Tam99] Koichiro Tamura. Treeexplorer manual. Available at: `http://evolgen.biol.metro-u.ac.jp/TE/TE_man.html`, July 1999.

Appendices

Appendix A

SPQR Algorithm

SPQR Algorithm Initialization and I/O

Input:
$m \times$n matrix A // original matrix
k // number of iterations, i.e., maximum number of columns to process
tol // desired accuracy

Output:
R // $k \times k$ triangular R_{11}
P // Permutation matrix
$nrmR22$ // norm of R_{22}

Initialization:
k = min (m, n, k)
$R[1:k, 1:k] = 0$
$colnrm[j] = \|A[:, j]\|, \quad j = 1, ..., n$ // norm of the columns of R_{22}
$P[j] = j, \quad j = 1, ..., n$

SPQR Algorithm

for $i = 1$ to k **do**
  // pivoting
  Determine an index p such that $conrm[p]$ is maximal
  $P[k] \leftrightarrow P[p]$
  $colnrm[k] \leftrightarrow colnrm[p]$
  $a = A[:, P[k]]$

  // special action for the 1st iteration
  if $k = 1$ **then**
    $R[1,1] = \|a\|$
    $q = a/R[1,1]$
  else
    // quasi-Gram-Schmidt step with reorthogonalization
    $b = a^T * A[:, P[1:k-1]]$
    Solve the system $R[1:k-1, 1:k-1]^T * R[1:k-1, k] = b^T$ for $R[1:k-1, k]$
    Solve the system $R[1:k-1, 1:k-1] * c = R[1:k-1, k]$ for c
    $q = a - A[:, P[1:k-1]] * c$
    $b = q^T * A[:, P[1:k-1]]$
    Solve the system $R[1:k-1, 1:k-1]^T * r = b^T$ for r
    Solve the system $R[1:k-1, 1:k-1] * c = r$ for c
    $q = q - A[:, P[1:k-1]] * c$

    // update R
    $R[1:k-1, k] = R[1:k-1, k] + r$
    $R[k,k] = \|q\|$

    // compute k^{th} column of Q
    Solve the system $R[1:k-1, 1:k-1] * c = R[1:k-1, k]$ for c
    $q = (a - A[:, P[1:k-1]] * c)/R[k,k]$
  end if

  // compute the k^{th} row of R_{12}
  if $k+1 \leq n$ **then** $r[k+1:n] = q^T * A[:, P[k+1, n]]$ **end if**

  // downdate column norms, compute frobenius norm of R_{22}
  $colnrm[j] = colnrm[j]^2 - r[j]^2, \qquad j = k+1, ..., n$, set negative downdates to 0
  $nrmR22 = sqrt(sum(colnrm[j])), \qquad j = k+1, ..., n$
  $colnrm[j] = sqrt(colnrm[j]), \qquad j = k+1, ..., n$

  // check for stopping criterion
  if $nR22 < tol$ **then** leave k **end if**
end for

Appendix B

Experimental Results Tables

Table B.1: Performance Tables. (a) Plant dataset timings, (b) Vertebrate dataset timings, and (c) Eukaryote dataset timings.

(a)	SPQR(A)	SPQR(A^T)	SVD
Alg. Time	115.54 sec	17.08 sec	725.54 sec
COSDIST Time	—	30.68 sec	17.12 sec
NEIGHBOR Time	—	9.16 sec	5.58 sec

(b)	SPQR(A)	SPQR(A^T)	SVD
Alg. Time	850.55 sec	6.78 sec	2443.62 sec
COSDIST Time	—	13.92 sec	3.37 sec
NEIGHBOR Time	—	12.56 sec	3.01 sec

(c)	SPQR(A)	SPQR(A^T)	SVD
Alg. Time	1.733 hrs	2.216 hrs	12.350 hrs
COSDIST Time	—	2.835 hrs	0.175 hrs
NEIGHBOR Time	—	0.003 hrs	0.001 hrs

Table B.2: Memory Use Table (in Megabytes)

(a)	SPQR(A)	SPQR(A^T)	SVD
Plants	4.431372	4.581984	9.462008
Vertebrates	26.913548	2.611772	29.000680
Eukaryotes	20.513492	20.477984	32.252480

Vita

Shakhina Pulatova was born in Tashkent, Uzbekistan on Febuary 16, 1982. She attended elementary and high school in Tashkent graduating with highest honors in 1999. She received a Bachelor of Science degree in Computer Science with a Mathematics minor from the University of Tennessee at Knoxville in 2002. She is currently pursuing her Masters degree in Computer Science at the University of Tennessee, Knoxville.