Multiple sequence alignment algorithms for the phylogenic analysis of chloroplast DNA.

By

Adam Rumbold (BSc.)

A dissertation submitted to the
School of Computing
in partial fulfilment of the requirements for the degree of

Bachelor of Computing with Honours

University of Tasmania

November, 2004

Declaration

I, Adam Rumbold, declare that this thesis contains no material which has been accepted for the award of any other degree or diploma in any tertiary institution. To my knowledge and belief, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Adam Rumbold

Abstract

The application of approximate string matching and alignment algorithms to either DNA or amino acid (protein) sequences is important for determining conserved regions, functional sites and to allow for multiple sequence alignments from which an evolutionary (phylogenic) tree may be inferred.

Global sequence alignment algorithms attempt to maximise the alignment score by placing gaps (which are seen as insertion/deletion evolutionary events) in either sequence so as to maximise the number of matching characters and minimise mismatches and gaps. The extension of traditional dynamic programming algorithms for aligning two sequences to aligning N sequences leads to a polynomial increase in the space and time complexity. Consequently many heuristic multiple sequence alignment algorithms, and improvements in the representation of a multiple sequence alignment, have been developed. This honours project has focussed on multiple sequence alignment algorithms, their processing and space requirements and the suitability of the alignments of samples of chloroplast DNA to further phylogenic analysis. Since the samples being used in the analysis are hypervariable, this research has also looked at algorithms capable of handling inversions in the DNA sequences (where a section of DNA has undergone the mutation of reversing)

Acknowledgements

I would like to thank my supervisors for their assistance and support throughout the year. Thanks must also go to Denise for her advice, proof reading skills and patience. Finally I would like to thank all the honours students for their many worthwhile distractions.

Table of Contents

Index of Tables

Table of Figures

1. Introduction

In recent years there has been an exponential increase in the amount of biological data being collected worldwide (see section 5.1). This wealth of data and the complexity of biological processes has nurtured a growing field of study in bioinformatics. Bioinformatics uses mathematical and informational techniques to solve biological problems, typically through computer programs or mathematical models. This research will focus on multiple sequence alignment techniques and their subsequent applications in inferring an evolutionary history (or phylogeny).

1.1 Biological Concepts

The cell is the basic structure of life and performs essential functions such as respiration, consuming nutrients and expelling metabolic waste. Within cells are DNA (deoxyribonucleic acid) molecules that 'encode' all the information necessary to produce the proteins essential to all cellular processes. It is for this reason that DNA is considered the 'blue-print' of life; and is recognised as distinguishing whether two living beings are biologically similar or distinct (Junior 2003).

The double helix structure of DNA was discovered by Crick and Watson in 1953. DNA consists of a double chain of simpler molecules called nucleotides. The nucleotides that comprise a strand of the double helix have a nitrogen base that can be of four types: adenine (A), cytosine (C), guanine (G) and thymine (T). These bases are the molecules that tie the double helix together. Each nucleotide consists of a sugar (eg deoxyribose in DNA), a phosphate and a base. The two strands comprising the double helix are complementary in the sense that adenine always bonds to thymine and cytosine always bonds to guanine. As such it is sufficient to know one strand to be able to deduce the other. The bonding between nucleotides form base pairs (bp), which is commonly used to specify DNA length.

As detailed by D'Antonio (D'Antonio 2003), proteins and nucleic acids are the main components of the biochemical processes of life. Proteins are molecules that determine both the shape and structure of a living cell as well as achieving the vital cellular functions such as respiration etc. A protein is a sequence comprised of a combination of 20 simpler molecules called amino acids. A sequence of three nucleotides (called a codon) code for a particular amino acid (see table 1), and the contiguous sequence of nucleotides that code for a particular protein is called a gene. In this sense a protein may be viewed as either a sequence of amino acids or equivalently a transcribed sequence of nucleotides that code for the specific amino acid sequence. The relationship between codons and amino acids is called the genetic code. Included in the genetic code are three special 'STOP' entries that indicate the end of a gene.

First		**Second position**			**Third**
position	G	A	C	T	**position**
G	Gly	Glu	Ala	Val	G
G	Gly	Glu	Ala	Val	A
G	Gly	Asp	Ala	Val	C
G	Gly	Asp	Ala	Val	T
A	Arg	Lys	Thr	Met	G
A	Arg	Lys	Thr	Ile	A
A	Ser	Asn	Thr	Ile	C
A	Ser	Asn	Thr	Ile	T
C	Arg	Gln	Pro	Leu	G
C	Arg	Gln	Pro	Leu	A
C	Arg	His	Pro	Leu	C
C	Arg	His	Pro	Leu	T
T	Trp	STOP	Ser	Leu	G
T	STOP	STOP	Ser	Leu	A
T	Cys	Tyr	Ser	Phe	C
T	Cys	Tyr	Ser	Phe	T

Table 1 Table of codons and their transcribed amino acid

Living organisms are made up of two types of cells prokaryote and eukaryote. Prokaryotes (eg bacteria) lack organelles (a structure with a specialised function) whilst most organisms including humans and plants are comprised of eukaryotic cells. "A eukaryotic cell has a nucleus, which is separated from the rest of the cell by a membrane. The nucleus contains chromosomes, which are the carriers of the genetic material- DNA. There are internal membrane enclosed compartments within eukaryotic cells, called organelles...which are specialised for particular biological processes." (Lopez)

Suspended in the cytoplasm of plant cells are plastids. These include amyloplast (used for starch storage), chromoplast (for pigment synthesis and storage), chloroplasts (for photosynthesis) and etioplasts (chloroplasts which have not been exposed to light).

1.2 Biological Sequences

DNA and protein sequences can be seen as long text strings over restricted alphabets. In the case of DNA there are four possible bases A, C, T and G that represent the alphabet, whilst amino acid sequences have an alphabet of 20 characters.

The comparison of two or more biological sequences can serve a number of purposes. Through the theory of evolution it is widely understood that gene sequences may have evolved from a common ancestral sequence. It is therefore of interest to study the evolutionary history of mutations (insertions and deltions of DNA as well as rearrangement mutations such as inversions and translocations) and other changes. The study of biological sequences can also be studied to locate regions of commonality, which may correspond to regions of similar structure or function.

The analysis of two or more sequences often occurs through the application of a sequence alignment algorithm and the measure of similarity that results. In aligning two or more sequences there are typically four possible operations that can be conducted on either sequence: insertion, deletion, substitution or transposition. In biology an insertion in one sequence can alternatively be seen as a deletion from the other. For this reason insertions or deletions are commonly referred to as *indels*. There are also a number of rearrangement events/mutations that can occur on biological sequences such as inversions (whereby a subsegment of DNA changes orientation), duplications (where a copy of a subsegment is inserted into the sequence), translocations (where a subsegment is removed and inserted in a different location) or a combination of the above.

When analysing nucleotide sequences single nucleotide polymorphisms (SNP's) are naturally occurring operations on nucleotides and are an important considerations in constructing realistic sequence alignment algorithms.

"They [SNP's] are most commonly changes from one base to another - transitions and transversions - but single-base insertions and deletions ("indels") are also SNP's…Transitions change a purine to a purine or a pyrimidine to a pyrimidine – A to G or C to T and vice versa. Transversions change a purine to a pyrimidine and vice versa – A or G to C or T; C or T to A or G. Even though there are twice as many possible transversions as transitions, transitions tend to be at least equal to transversions in frequency, and often more prevalent." (Gibson & Muse 2002)

1.3 Chloroplast DNA

Chloroplast DNA (cpDNA) is a coding region within plant cells, with many chloroplast DNA genes encoding proteins that are involved in photosynthesis. Depending on the species, chloroplast DNA has a size ranging from 110 000 bp to 160 000 bp (Blake). Samples of cpDNA have been useful in studies that take the geographic distribution of species into account when inferring an evolutionary history since cpDNA variation is correlated more with geographic distribution than with morphological species (Freeman et al. 2001). Chloroplast DNA is useful in studies because it is inherited maternally and is considered "a single, non-recombining unit of inheritance." (Schaal et al. 1998)

1.4 Sequence Alignment

Sequence alignment and specifically multiple sequence alignment (MSA) is of fundamental importance within bioinformatics. By analysing the similarities and differences of either nucleotide or amino acid sequences it is possible to infer structural, functional or evolutionary relationships between the sequences being studied (Baxevanis & Ouellette 1998).

The problem of aligning two sequences can be described as follows. Given two sequences of characters and a scoring system that ascribes scores to two matching characters, two mismatching characters and penalties for gaps, the aim of the sequence alignment problem is to produce a pairing of characters from one sequence with the second sequence so that the total score is optimal. In pairing characters, gaps can be inserted at any position in the sequences; however the order of characters in each sequence must be preserved. A pairwise alignment is an alignment of two sequences and if there are more than two sequences the problem is called multiple sequence alignment (MSA).

In aligning sequences there are two types of alignment: global and local. Global alignments attempt to align sequences over their entire length, whilst local alignments construct the best alignment of segments of the sequences that exhibit a high density of matches, ignoring the remaining regions of the sequences. Table 2 below shows examples of global and local alignments.

Global Alignment	Local Alignment
ACTGATTA ||| ||| ACT--TTA	ACTGATTA |||| --TGAT--

Table 2 Global and local alignment comparison

Multiple sequence alignment algorithms are global however a number use local alignment algorithms to anchor the global alignment.

As detailed in *Bioinformatics* (Baxevanis & Ouellette 1998, pp. 145-71) when analysing two or more sequences the similarity between these sequences is an observable quantity (for example a percentage identity score). The similarity score or scores may then be used to infer homology. Based on the similarity of two or more sequences it may be concluded that the sequences share a common evolutionary history and therefore are homologous.

2. Sequence Alignment

2.1 Introduction

Approximate string matching has been an important tool in computational biology since small variations within DNA or protein sequences are common. Biologists often require searching the online databases with particular sequences so as to detect homologous sequences in other species. These searches allow biologist to analyse the similarities and differences of either nucleotide or amino acid sequences to aid in inferring structural, functional and/or evolutionary relationships between the sequences (Baxevanis & Ouellette 1998). It is also necessary to align sequences in the first stage of the process of creating phylogenic trees.

There is no such thing as a single best alignment with biological sequences since optimality always depends upon the assumptions made in the alignment algorithm such as the penalty ascribed to gaps or particular substitutions as well as the mutational events the algorithm is capable of handling. These decisions must be carefully considered according to the observed variations in the DNA or protein samples being considered (Baxevanis & Ouellette 1998). Furthermore, most sequence comparison methods use sequence alignment algorithms that inherently "assumes conservation of contiguity between homologous segments" (Vinga & Almeida 2003). This assumption is at odds with genetic recombination such as translocations and inversions. Stochastic modeling of sequences using hidden Markov models (Muckstein, Hofacker & Stadler 2002) and recently alignment-free sequence comparison methods (Vinga & Almeida 2003) are new techniques to overcome this.

2.2 Dot Plots

Dot Plots are an effective visual representation of the similarities between two sequences. A Dot Plot is a 2-dimensional image whereby each axis represents one of the two sequences being compared. A small window size is set, along with a rule that determines when two sequence windows are similar. When one window in one sequence is similar under the rule to another window from the other sequence, a dot or small diagonal line is drawn. This gives an overall representation of the similarity globally and locally of the two sequences. That is global similarity is represented by a diagonal line from the top left to the bottom right of the dot-plot, whilst local similarity is shown by smaller diagonal lines within the dot-plot (Maizel & Lenk 1981). Dot plots of the data set examined are shown in section 7.1.

2.3 Sequence Alignment Techniques

Most formulations of sequence alignment consist of an objective function which assigns a score to each possible alignment of the sequences. The computational problem is to find an alignment which either optimizes the objective function through heuristic techniques, or guarantees an optimal score such as through a dynamic programming technique.

2.3.1 Dynamic Programming

Dynamic programming algorithms were initially developed to calculate the minimal edit distance between two sequences (Needleman & Wunsch 1970; Sellers 1974). The first dynamic programming algorithm to compute the edit distance and search for a pattern sequence within a text was developed by Sellers in 1980 (Sellers 1980). Many variations have been rediscovered and both theoretical and practical improvements have since been made (Chang & Lawler 1994; Chang & Marr 1994; Galil & Park 1990; Ukkonen 1985). These improvements have yielded an average case $O((kn)/\sqrt{\sigma})$ and worst case $O(kn)$ algorithms (Navarro 2001), where σ is the size of the alphabet being considered and n is the size of the largest sequence.

Dynamic programming algorithms are particularly flexible in handling different distance functions, although they are not the most efficient algorithms in general. Dynamic programming routines guarantee the mathematically optimal alignment, and can easily be generalised to optimally align N sequences, however they process in $O(L^N)$ time where L is the length of the longest sequence and hence are unsuitable for any reasonable number of sequences.

Dynamic programming is a method of constructing a solution gradually by using recurrences. In calculating the minimum edit distance between two sequences A and B, we first introduce the notation D(i,j) to represent the edit distance between the first i characters of A and the first j characters of B. The minimal edit distance of A and B (as defined in section 3.1) is D(n,m) and is calculated by first computing D(i,j) where $i<n$ and $j<m$. This is achieved through the construction of a dynamic programming matrix and by a simple recurrence (Gusfield 1999).

The edit-distance metric scores a 1 for each mismatch or inserted gap and a 0 for a match. To calculate the optimal alignment of A and B (using the simple edit-distance metric) a matrix $D_{0..n,\ 0..m}$ is first constructed, where each element $D_{i,j}$ represents $D(i,j)$ and $D_{n,m}$ represents the minimum edit distance between A and B. The matrix is firstly initialised such that $D_{i,\ o} = i$ and $D_{0,\ j} = j$ since this represents the edit distance between strings of length i or j respectively and the empty string which is equivalent to i/j deletions from the non-empty string. Then all the matrix values are calculated according to the formula:

$D_{i,\ j} = \min(D_{i-1\ ,\ j-1} + d(A_i\ ,\ B_j)\ ,\ D_{i-1\ ,\ j} +1\ ,\ D_{i\ ,\ j-1} +1)$ where $d(A_i\ ,\ B_j)$ is 1 if $A_i \neq B_j$ and 0 otherwise.

To find the optimal global alignment requires a traceback through the matrix, and to achieve this it is necessary to know the path by which each new value in the matrix was calculated. That is, in finding $D_{i,\ j}$, we also store a pointer to cell $D_{i-1,\ j}$ if $D_{i-1\ ,\ j}$ +1 was minimal, and/or a pointer to cell $D_{i,\ j-1}$ if $D_{i\ ,\ j-1}$ +1 was minimal and/or a pointer to cell $D_{i-1,\ j-1}$ if $D_{i-1\ ,\ j-1} + d(A_i\ ,B_j)$ was minimal. By storing pointers indicating the direction from which the minimal value(s) were found an optimal alignment can simply be found through following *any* path from $D_{n\ m}$ to $D_{0,0}$.

D(i ,j)	0	1 (A)	2 (G)	3 (A)	4 (P)	5 (E)
0	0	← 1	← 2	← 3	← 4	← 5
1 (G)	↑ 1	↖ 1	↖ 1	← 2	← 3	← 4
2 (R)	↑ 2	↑ ↖ 2	↑ ↖ 2	↖ 2	← ↖ 3	← ↖ 4
3 (A)	↑ 3	↖ 2	←↑ ↖ 3	↖ 2	←↖ 3	←↖ 4
4 (P)	↑ 4	↑ 3	←↑ 4	↑ 3	↖ 2	↖ 4
5 (E)	↑ 5	↑ 4	↖ 4	↑ 4	↑ 3	↖ 2

Figure 1 **Dynamic programming matrix example**

In the above example (figure 1) the edit distance between "GRAPE" and "AGAPE" is found to be two and to find the optimal alignment a traversal back through the matrix following the direction of the pointers. When cell (2,2) is reached a choice between two optimal alignments must be made (either a substitution of G to R or a deletion of "R" from "GRAPE". The two possible global alignments of the two sequences is shown below, both have an edit distance of 2:

Unaligned	Aligned	
`AGAPE`	`AG-APE`	`AGAPE`
`GRAPE`	`-GRAPE`	`GRAPE`

The above method can be easily altered to find localised regions of similarity by allowing for sequences that match the pattern to begin anywhere within the text. A locally matching region is found by selecting those areas within the matrix where the score is above some similarity threshold. The basic routine outlined above can also be easily extended to handle various other distance functions (include affine-gap penalties where the penalty ascribed to x insertion or deletions is given by a $\gamma + \lambda x$ where γ is the gap-initiation cost and λ is the gap-extension cost).

2.3.2 Anchoring

Anchoring methods make use of a local alignment algorithm to first find regions of high similarity within the sequences, and then solve the problem of finding the best chain of the local alignments around which a global alignment is fixed. A local alignment can be represented by a series of diagonal pointers in the dynamic programming matrix. To chain two local alignments we require that a monotonic conservation map can be created. That is, if local alignment A is defined by (x_{1A} , y_{1A}) and (x_{2A} , y_{2A}) and local alignment B is defined by co-ordinates (x_{1B} , y_{1B}) and (x_{2B} , y_{2B}) we require that $x_{2A} < x_{2B}$ and $y_{2A} < y_{2B}$. A typical approach used by anchoring algorithms are to select the optimal chain of local alignments and then to use a global alignment approach in the areas between each local alignment.

There are a number of global pair-wise alignment algorithms which chain together regions of high similarity such as CHAOS, BLAST and LAGAN, as well as multiple sequence alignment algorithms DiAlign and Mavid.

2.4 Sequence Alignment Algorithms

2.4.1 NEEDLEMAN-WUNSCH

The Needleman-Wunsch (NW) algorithm (Needleman & Wunsch 1970) was the first algorithm to determine an optimal global alignment of two sequences. The algorithm developed a dynamic programming approach to calculating the mathematically optimal alignment over all residues of the sequences.

The original method is rarely used due to its cubic runtime (Gusfield 1999, p. 234), however through numerous improvements the run time has been improved (see section 2.3.1).

2.4.2 CHAOS (Chain of Scores)

The CHAOS algorithm (Brudno et al. 2003) is a heuristic local alignment algorithm that builds a global alignment using a set of local alignments called anchor points. CHAOS can be used as a stand-alone local sequence alignment program or as a pre-processing step for a global alignment algorithm (eg LAGAN, SLAGAN).

The algorithm works by chaining *seeds* – a pair of words of length k that has at least n identical characters. Two seeds, s_1 and s_2 can be linked together if: the indices of s_1 are higher than those of s_2 in both sequences and the two seeds are 'near' to each other. Two seeds satisfy this second condition if they are within a region defined by a gap score and distance cut-off, as illustrated in figure 2.

The seeds are located using a variation of the Aho-Corasick algorithm (Aho & Corasick 1975) which uses a variation on the trie data structure, the threaded-trie (T-trie). The trie is a tree like data structure that stores all the k-mers of one sequence, with one node for each common prefix. The T-trie differs in that nodes representing the string $w_1..w_p$ also store back-pointers to the node containing the string $w_2..w_p$.

The algorithm begins by inserting all of the k-mers of one sequence (database sequence) into the T-trie. The root of the T-trie is made the current node, and for every letter of the other (query) sequence:

1. If the current node has a child node corresponding to the letter under consideration this node is made the current node and any seeds stored in it are returned.
2. Otherwise make the node pointed to by the back-pointer the current node and return to step 1.

To keep a track of the available seeds to chain to, a probabilistic data structure called a skip list is used. A distance, D, specifies the maximal distance for which two seeds must be under to be chained together (see figure 2). Any seeds generated whilst examining the last D base pairs of the query sequence are stored in the skip-list. The seeds are ordered by the difference of its indices in the two sequences (diagonal number). For each seed *s* found at the current location, its diagonal number is found and a search of the skip-list is made to find any seeds in the skip-list whose diagonal number is within the allowable gap cut-off. This finds all possible previous seeds that *s* can be chained to, and the highest scoring chain is chosen.

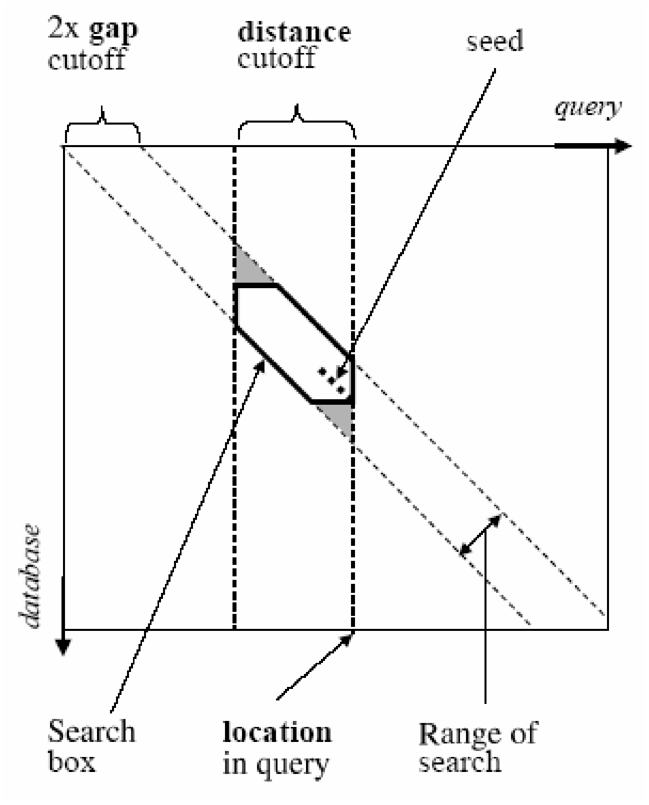

Figure 2 CHAOS illustration from (Brudno et al. 2003)

2.4.3 LAGAN

Limited Area Global Alignment of Nucleotides — Lagan — is a fast heuristic pairwise alignment algorithm developed by Michael Brudno (Brudno et al. 2003). The algorithm first uses the CHAOS algorithm to find the set of all local alignments. These alignments are then weighted and the optimal chain (based upon the longest increasing subsequence algorithm) is chosen, with each local alignment in this chain an *anchor* to the global alignment. LAGAN then calls CHAOS recursively between two anchors when they are more than a threshold apart to overcome sparse regions. A Needleman-Wunsch like global alignment algorithm is then called in the areas between two anchors, and in a restricted area surrounding the anchors (ie. those cells within a distance of *r* from the anchor).

LAGAN uses memory proportional to the largest area between two local alignments plus the memory needed to hold the alignment (Brudno et al. 2004).

2.4.4 SLAGAN

Shuffle-Lagan (SLAGAN) is an extension of the LAGAN algorithm to handle rearrangement mutations (such as inversions or translocations) as well as indels. The algorithm relaxes the restriction that both indices from two local alignments must be strictly increasing in order to chain the local alignments so that only one of the indices must be strictly increasing. This relaxation allows local alignments that align a region of one sequence with a region in the other sequence whose orientation is reversed (ie. an inversion) so long as order is preserved in at least one of the sequences.

3. Multiple Sequence Alignment

3.1 Introduction

All MSA algorithms require an objective function to determine the relative quality of each possible multiple sequence alignment. The sum-of-pairs (SP) objective function scores a MSA by the sum of the induced pair-wise alignments. For example, considering the following simple MSA:

```
A: ACT-GGT
B: -CTGGGT
C: CCTGG-T
```

This alignment induces the following pair-wise alignments

```
A: ACT-GGT
B: -CTGGGT

A: ACT-GGT
C: CCTGG-T

B: -CTGGGT
C: CCTGG-T
```

The sum of pairs objective function then scores the MSA M as S(M) = $\sum_{k<l} s(k,l)$ where s(*k,l*) is the pair-wise score of sequences k and l from MSA M. If a guide tree is known the weighted sum of pairs may be used to down-weight induced alignments which are less related than those closely related.

Multiple sequence alignment has been shown to be an NP-complete problem using the sum of pairs objective function (Bonizzoni & Vedova 2001; Wang, L. & Jiang 1994). All the multiple sequence alignment algorithms that are practical for realistic data sets and reflect real-world models of evolution are heuristic (Vision & McLysaght 2004), and as such they do not guarantee an optimal alignment.

Multiple sequence alignment (MSA) algorithms can be classified into three classes: iterative, progressive and exact. The most widely used heuristic algorithms are based on the progressive alignment of sequences to create a multiple sequence alignment (Vision & McLysaght 2004). Typically however, expert knowledge is needed to review and for manual editing of a MSA to produce a 'good' alignment (Baxevanis & Ouellette 1998, p. 173; Thompson, J., Higgins & Gibson 1994).

A list of MSA applications is provided in section 5.2.2

3.2 Concepts

Progressive alignment methods are the most commonly used and have the advantage of speed and simplicity (Notredame 2002). Progressive alignment successively aligns pairs of sequences using pairwise alignment algorithms (such as Needleman-Wunsch etc). Progressive alignment algorithms differ in several key ways: the way they choose the order in which to do the alignment, if they involve the alignment of a single sequence to a single growing alignment or if subfamilies are built up leading to alignments of alignments, and the method of aligning and scoring sequences or alignments against existing alignments. The most important heuristic used in progressive alignment algorithms is to align the most similar sequences first (those with the smallest edit distance). Progressive sequence alignment algorithms are sensitive to the order of the pairwise alignments which is determined solely by alignments of only two sequences at a time (Morgenstern, Dress & Wener 1996). This has been addressed recently by using a travelling salesman approach to determine the order of alignments (Chantal & Gaston 1999).

Feng and Doolittle (1996) developed the key idea that the strings with minimal edit distance are most likely to be from species that have most recently diverged, and therefore these strings provide the most 'reliable' information contained in the multiple sequences. The method of Feng and Doolittle preserves gaps within these closely related sequences and progressively aligns sequences with greater edit distances.

Iterative methods produce a MSA and then refine the alignment through stochastic or deterministic iterations (or cycles). Hidden Markov Models, and other statistically-based methods, have been used to attempt to associate a probability value to an alignment, and to encapsulate some of the known evolutionary information into the MSA.

3.3 Multiple Sequence Alignment Algorithms

3.3.1 Aligning Alignments Exactly

Kececioglu and Starret (Kececioglu & Starrett 2004) recently published the first algorithm to align alignments exactly (AAE) using linear gap costs and under the sum of pairs objective function. “Aligning Alignments is the problem of finding an optimal alignment of the columns of two multiple sequence alignments under the sum-of-pairs objective with linear gap-costs. The *sum-of-pairs* objective scores a multiple alignment by the sum of the scores of the two-sequence alignments induced on all pairs of sequences. With *linear gap-costs* a run of either x insertions or deletions costs $\gamma + \lambda x$ where γ is the gap-initiation cost and λ is the gap-extension cost.”(Kececioglu & Starrett 2004) Although an NP-complete problem, the algorithm was published with a number of speed-ups leading to a linear run time.

The algorithm is based upon a dynamic programming routine where each alignment is treated as a sequence of columns. When aligning an alignment A of k rows and m columns to an alignment B of l rows and n columns an $m+1$ by $n+1$ grid structured graph is constructed. The graph is traversed in row-major order until the final cell (m,n) has been calculated. Costs within the dynamic programming table are calculated according to the cost of composing a column to the current alignment.

To calculate the number of gaps initiated over all the sequences by composing a column to an alignment it is necessary to introduce the notion of a shape. A shape is an ordered partition of the rows (of both alignments) which indicate the order in which each rows final character finishes. For example the shape {(3)(0)(1 2)} indicates that the alignment being considered has row 3 finishing first (followed by gaps), then row 0, with the sequences in positions 1 and 2 having their final character last. That is, row 3 *underhangs* rows 0,1 and 2, whilst row 0 *overhangs* row 3 in the alignment. Each cell in the table can have a number of shapes and corresponding scores for the optimal alignment ending in that shape.

The column to be added to the current alignment is determined through the position in the table and the direction of the cell under consideration. If we let the set of shapes at a particular entry (i, j) in the table be denoted by $S(i,j)$. When at position (i, j) in the dynamic programming table and considering shape $s \in S(i, j)$ the calculation of the score at the adjacent cells is detailed below:

$(i, j+1)$ = The cost of composing a column consisting of gaps in alignment A and characters from column $j+1$ in alignment B to the shape *s*.
$(i+1, j)$ = The cost of composing a column of characters from column $i+1$ in alignment A and gaps in alignment B to the shape *s*.
$(i+1, j+1)$ = The cost of composing a column of characters from column $i+1$ in alignment A and characters from column $j+1$ in alignment B to the shape *s*.

Using the following notation:
$A[i, j]$ = the character at row *i* and column *j* of alignment A
$A[j]$ = column *j* from alignment A
$S(i, j)$ = the set of shapes at entry (i, j) in the dynamic programming table
$C(i, j, s)$ = the cost of the alignment of prefixes $A[1:i]$ and $B[1:j]$ that ends in shape s.
$S(i,j) o\ c$ = the set of shapes obtained by composing column *c* to the set of shapes $S(i,j)$.

"To find an optimal alignment of A and B, the subproblem we solve is to determine for a given shape *s* and indices $0 <= i <= m$ and $0 <= j <= n$, the cost of an optimal alignment of the prefixes $A[1: i]$ and $B[1: j]$ that end in shape *s*."(Kececioglu & Starrett 2004)

Starting from cell (0,0) the shapes and scores contained in the table are calculated using the following recurrences:

$$S(i,j) = \begin{cases} ((1,\ldots,k,k+1,\ldots,k+l)), i = 0 \;\&\; j = 0 \\ S(i-1,j)o(A[i],-) \\ \cup\, S(i,j-1)o(-,B[j]) \\ \cup\, S(i-1,j-1)o(A[i],B[j]), otherwise \end{cases}$$

To count the number of gaps initiated the following notation is introduced. For rows *p* and *q* in an alignment of shape *s* and column *c:*

- ${}_q s^p$ if, and only if, *p* overhangs *q* in the alignment
- ${}^p s_q$ if, and only if, *p* underhangs *q* in the alignment
- ${}_q c^p$ if, and only if, *p* has a letter and *q* has a gap in column *c*
- ${}^p c_q$ if, and only if, *q* has a letter and *p* has a gap in column *c*

Using the notation (a,b) to denote a column where a is either a column from alignment A or a column of gaps('-') and b is either a column from alignment B or a column of gaps. The total number of gaps initiated by composing a column (a,b) onto an alignment that has shape *s* is:

$$g(a,b,s) = \sum_{\substack{p \in A \\ q \in B}} (({}_q(a,b)^p \;\&\; \overline{{}_q s^p}) \,||\, ({}^p(a,b)_q \;\&\; \overline{{}^p s_q}))$$

where $\overline{}$ denotes logical negation (NOT), || denotes logical OR and the & denotes logical AND. In the above summation a true value maps to 1 and false maps to 0.

The recurrences for C*(i,j,s)* is based on the fact that an optimal alignment of A[1:*i*] and B[1:*j*] ending in shape *s* must have a final column *c* and removing this column one must be left with an optimal alignment ending in shape $\tilde{s}$ such that $\tilde{s}\ o\ c = s$. So for 0 <= *i* <= m and 0 <= *j* <= n

$$C(i,j,t) = \min \begin{cases} \min\limits_{\substack{s \in S(i-1,j) \\ so(A[i],-)=t}} \begin{Bmatrix} C(i-1,j,s) + \\ \gamma g(A[i],-,s) + \lambda l \left| A[i] \right| \end{Bmatrix} \\ \min\limits_{\substack{s \in S(i,j-1) \\ so(-,B[j])=t}} \begin{Bmatrix} C(i,j-1,s) + \\ \gamma g(-,B[j],s) + \lambda k \left| B[j] \right| \end{Bmatrix} \\ \min\limits_{\substack{s \in S(i-1,j-1) \\ so(A[i],B[j])=t}} \begin{Bmatrix} C(i-1,j-1,s) + \\ \gamma g(A[i],B[j],s) + \sum\limits_{\substack{p \in A \\ q \in B}} \sigma(A[p,i],B[q,i]) \end{Bmatrix} \end{cases}$$

Where $|c|$ is the number of characters in the column, and $\sigma(a,b)$ is the substitution cost for a match/mismatch of a character *a* from alignment A with a character *b* from alignment B.

To illustrate the basic principles of the algorithm consider the following example of aligning two alignments of two sequences each:

Sequence number	**Alignment A**	**Sequence number**	**Alignment B**
0	`ATG`	2	`ACTG`
1	`ATG`	3	`-CTG`

A dynamic programming table is constructed (table 3) starting with the ‘flat’ shape of score 0. From this initial cell (0,0) the shapes and scores for cells (0,1), (1,0) and (1,1) are calculated. The calculations then move onto cell (0,1) and for all shapes in this cell the resulting shapes and scores for cells (0,2), (1,1) and (1,2) are calculated. This process continues until all cells in the table have been filled.

((1234)) = 0	((013)(2)) = -42	((01)(23))= -96	((01)(23))=-120	((01)(23))=-144
((23)(01)) =-84	((3)(012))=-28 ((3)(2)(01))=-126 ((3)(01)(2))=-126	((0123))=-42 ((23)(01)=-180 ((01)(23))=-112	((0123))=-96 ((23)(01))=-204 ((01)(23))=-126	((0123))=-120 ((23)(01))=-228 ((01)(23))=-150
((23)(01))=-108	((3)(012))=-84 ((3)(2)(01))=-82 ((3)(01)(2))=-150	((0123))=-28 ((23)(01))=-126 ((01)(23))=-166	((0123))=-38 ((23)(01))=-180 ((01)(23))=-112	((0123))=-96 ((23)(01))=-204 ((01)(23))=-122
((23)(01))=-132	((3)(012))=-108 ((3)(2)(01))=-106 ((3)(01)(2))=-174	((0123))=-82 ((23)(01))=-112 ((01)(23))=-190	((0123))=-28 ((23)(01))=-122 ((01)(23))=-166	((0123))=-34 ((23)(01))=-180 ((01)(23))=-112

Table 3 Aligning Alignments example

The optimal alignment is then constructed through a traceback procedure similar to that used in the NW dynamic programming algorithm. The shape with the highest score at cell (*m,n*) is first selected (see shaded cell at position (3,4)). The corresponding shape ((0123)) indicates that characters from both alignment A and B are in the final column of this alignment. This, in-turn, indicates that a substitution column must have been composed to give this shape and score. Consequently a move to cell (*m-1, n-1*) can be made (ie. back one column in each alignment to cell (2,3)). From cell (2,3) we consider all shapes in S(2,3) and determine the shape which under composition of a substitution column (A[3],B[4]) gives a score of -34. This calculation yields shape ((0123)) of score -38. This process continues until the cell (0,1) is reached where the shape is ((013)(2)). This shape indicates that only row 2 has a character in the final column and hence that gaps must have been inserted in alignment A (rows 0 and 1). This brings us to the original cell (0,0).

By following the path through the table and using the direction and positions in the table to determine the column of composition the optimal alignment is constructed as shown below.

```
-ATG
-ATG
ACTG
-CTG
```

The processing time and memory requirements are dependant upon the number of shapes, which in turn is dependant upon the gap structure of each alignment. Kececioglu and Starret developed two techniques (dominance pruning and bound pruning) which reduce the number of shapes added to the dynamic programming table. Bound pruning is very effective, with a large proportion of the table containing empty shape lists (Kececioglu & Starrett 2004).

3.3.2 Partial Order Alignment

A directed acyclic graph (DAG) representation of aligned sequences using a partial order graph has allowed for the development of an efficient alignment algorithm called Partial Order Alignment (POA) (Lee, Grasso & Mark 2002).

During progressive multiple sequence alignment algorithms there is the problem of incorrect scoring due to artifactual gap counts. When constructing a MSA through aligning a sequence to the current MSA, the MSA is first reduced to a *consensus sequence* or *profile.* This reduction results in a loss of information which inturn can lead to incorrect gap costs.

Artifactual gap counts are a legacy of aligning a sequence(s) to the profile of an alignment as illustrated in the simple example adapted from Lee et. al (Lee, Grasso & Mark 2002) below:

Consider the sequences:

```
A)                TGACTCGATATATCG
B)                CAGTCCGATAAGTCGTATCG
C)                CAGTCCGATAAGTCGTATCG
```

A possible global alignment of sequences A and B is shown below, with its corresponding profile sequence (with sequence C shaded along its length):

```
Alignment 1:
-----TGACTCGATA-----TATCG
CAGTC-----CGATAAGTCGTATCG
  *              ^
```

```
profile 1:
CAGTCTGACTCGATAAGTCGTATCG
```

The problem of artifactual gap costs can be seen when examining the gaps in the two shaded columns of the alignment, and the effect of adding sequence C to this current alignment. Whilst the gap above the ^ is a true gap, the gap above the * is an artefact of the alignment, as another equivalent alignment demonstrates:

```
Alignment 2:
TGACT-----CGATA-----TATCG
-----CAGTCCGATAAGTCGTATCG
profile 2:
TGACTCAGTCCGATAAGTCGTATCG
```

If sequence C was aligned to alignment 1 there would be a 5 residue gap penalty, however this does not occur in alignment 2.

The partial order alignment represents alignments 1 and 2 as the same partial order graph. Sequence A and B are represented as a graph in a process shown below adapted from Lee et. al (Lee, Grasso & Mark 2002):

a) The standard row-column representation of the sequence alignment

```
TGACT-----CGATA-----TATCG
-----CAGTCCGATAAGTCGTATCG
```

b) Each sequence is represented as a linear graph with each character a node and the order preserved.

T→G→A→C→T→C→G→A→T→A→T→A→T→C→G

C→A→G→T→C→C→G→A→T→A→A→G→T→C→G→T→A→T→C→G

c)The nodes/characters from each sequence that align are fused to create a graph structure as shown below

$$\begin{tikzcd}
T \arrow[r] & G \arrow[r] & A \arrow[r] & C \arrow[r] & T \arrow[dr] \\
& & & & & C \arrow[r] & G \arrow[r] & A \arrow[r] & T \arrow[r] & A \arrow[rrrrrr] \arrow[dr] & & & & & & T \arrow[r] & A \arrow[r] & T \arrow[r] & C \arrow[r] & G \\
C \arrow[r] & A \arrow[r] & G \arrow[r] & T \arrow[r] & C \arrow[ur] & & & & & & A \arrow[r] & G \arrow[r] & T \arrow[r] & C \arrow[r] & G \arrow[ur]
\end{tikzcd}$$

The MSA using POA algorithm is essentially a progressive dynamic programming algorithm. The algorithm uses a standard Needleman-Wunsch routine, however instead of aligning a profile sequence to another sequence, the partial order (alignment) is used. To cope with multiple sequences without going into an N dimensional space each bifurcation in the graph structure becomes another dynamic programming plane whose cells must be considered when tracing back through the grid to determine the optimal alignment.

3.3.3 ClustalW

ClustalW (Thompson, J., Higgins & Gibson 1994) is a progressive multiple sequence alignment algorithm that improves the sensitivity through selective weighting of sequences and substitution scores. ClustalW performs a pairwise alignment on all the sequences in order to construct a binary tree of their evolutionary relationship. This is then used to build a MSA by aligning the most recently diverged sequences first. ClustalW creates N/2 alignment profiles, which are then aligned to each other resulting in N/4 profiles. This process is continued until all the sequences have been aligned.

ClustalW has been widely used by biologists, as the algorithm was developed to try and overcome local minimum problems and incorrect alignments through the choice of alignment parameters. ClustalW dynamically varies the alignment parameters according to the sequence order and residue position so that gap insertions are penalised more heavily in highly conserved areas than in variable regions.

There are a number of factors influencing the changes in parameters (specifically the gap opening penalty or GOP) made by ClustalW throughout the alignment procedure. When a mismatch is scored the average cost is used as a scaling factor in an attempt minimise the dependance on the choice of weight matrix. Closely related sequences (based on the percentage identity score) use a higher GOP, with less similar sequences having a linearly reducing GOP. Longer sequences increase the alignment scores, so the GOP is logarithmically scaled based on the length of the shorter sequence being aligned. These three factors alter the GOP so that for sequences of length N and M:

GOP = {GOP+log[min(N,M)}*(average residue mismatch score)*(% identity scaling factor)

The gap extension penalty - GEP, is modified according to the difference in lengths of the two sequences. If there is a large difference in lengths of the two sequences, then the GEP is increased to limit the number of long gaps in the shorter sequence. So that:

$$GEP = GEP*[1.0+ |\log(N/M)|]$$

The GOP and GEP are also then altered throughout the alignment procedure depending upon the position. If there is a gap at a position then the GOP and GEP are lowered so that:

GOP = GOP*0.3*(number of sequences without a gap / no of sequences)
GEP = GEP*0.5

If the position being considered does not contain any gaps but a gap is within 8 positions then the GOP is increased so that:

GOP = GOP*{2 + [(8- distance from gap)*2]/8}

ClustalW also alters the GOP based on amino acid properties of the sequences. All of these factors, and the speed of the algorithm have led to arguably the most effective MSA algorithm for biologists today.

3.3.4 DiAlign

DiAlign (Morgenstern 1999; Morgenstern, Dress & Wener 1996) is an anchor based algorithm that works by aligning gap-free segments of variable lengths from both sequences. The segments under comparison appear as a diagonal on a dot-plot, and are the basis of the diagonal alignment – DiAlign algorithm.

To compute the best alignment, DiAlign finds the maximal scoring set of consistent diagonals. Diagonals D_1 and D_2 are consistent diagonals if the positions (residues) aligned to each other in D_1 precede those aligned in D_2, or vice-versa (see section 2.3.2). To determine the significance of a diagonal the following formulae are used:

For a fixed diagonal D, of length *l* and containing *m* matches, with *p* the probability of a point in the dot plot (ie. 0.25 for nucleic acids).

$$P(l,m) = \sum_{i=m}^{l} \left(\frac{l}{i}\right) p^{i} (1-p)^{l-i}$$

Then using the negative logarithm, the weighting of D is defined as:

$$w(D) = \begin{Bmatrix} -\ln(P(l,m)), E(l,m) > T \\ 0, otherwise \end{Bmatrix}$$

A high scoring weight indicates that a random diagonal of length *l* is unlikely to contain as many as *m* matches by chance. DiAlign scores highly for short segments with a large number of matches or for longer segments with fewer matches, so long as the segment is long enough.

For a set of diagonal $D_{1,}$ $D_{2,}$ $D_{3,...,}$ D_k the score is defined as:

score(*i, j)* := $\sum_{i=1}^{k} w(D_i)$. and

$$\text{prec}(i, j) := D_k = \begin{cases} prec(i, j-1), if \quad score(i, j) = score(i, j-1) \\ prec(i-1, j), if \quad score(i, j-1) < score(i, j) = score(i-1, j) \\ D_{i,j}, if \quad score(i, j-1), score(i-1, j) < score(i, j) = \sigma(D_{i,j}) \end{cases}$$

A consistent set of diagonals with a maximum score is a maximum alignment. The maximum alignment of two sequences is then calculated using a dynamic programming algorithm based upon maximising the score of the alignment of prefixes of both sequences. This is achieved through the following definitions and recurrences:

σ (D) := the maximum sum of weights up to and including diagonal D (D must have a positive weight).

π (D) := the diagonal preceding D.

score(i, j) := the score of the maximum alignment of $A_{1...i}$ and $B_{1..j}$

So, σ (D) = score (i-k-1, j-k-1) + w(D) and

π (D) = prec(i-k-1, j-k-1)

Now a recursive formulation of the *score* is given by:

score(i, j) =max{score(i ,j-1) , score(i-1, j) , σ ($D_{i,j}$) } , where $D_{i,j}$ is any diagonal ending at point (i, j) that satisfies σ ($D_{i,j}$) = max{σ (D): D ends at point (i, j)}.

To generalise DiAlign to N sequences "…we try to select a consistent set of diagonals with a maximal sum of weights. However, now diagonals originate from all the 1/2N(N-1) possible pairwise sequence comparisons."(Morgenstern, Dress & Wener 1996). DiAlign sorts diagonals according to their weights, irrespective of which sequence the diagonal originates from. The set of all diagonals from all the maximal pairwise alignments is sorted by weights and the overlap score (favouring diagonals that occur over multiple sequences), and diagonals are added individually in order to the multiple alignment so long as the diagonal being considered is consistent.

Recently CHAOS has been used to find anchor points that enables a speed-up of DiAlign by 1-2 orders of magnitude (Brudno et al. 2003). DiAlign is advantagous in that it can be applied to sequences that are both globally and locally related sequences.

3.3.5 Mavid

Mavid is a progressive global multiple alignment algorithm based upon the alignment algorithm AVID (Bray, Dubchak & Pachter 2002) and using maximum likelihood phylogenic techniques.

The sequences to be aligned are first aligned in a pairwise manner to produce a guide tree. This guide tree is used to determine the order in which sequences are aligned. The key difference in the MAVID algorithm to other progressive alignment algorithms is that instead of aligning two alignments directly through a consensus sequence, MAVID firstly infers ancestral sequences using standard phylogenetic models and then uses the alignment of these ancestral sequences using AVID to dictate the actual alignment.

AVID is a global alignment algorithm, that finds local alignments (matches) by concatenating the two sequences in question and then solving the maximal repeated substring problem using suffix trees. Maximal matches between two sequences are subsequences (one from each sequence) whose neighbouring bases are mismatches. AVID then uses a recursive approach to anchor and align the sequences based on the set of maximal matches. The anchors are selected by scoring each match based on its length and the alignment score (found through a variation of the Smith-Waterman local alignment algorithm) of the neighbouring regions 10 bp on either side of the match. The set of matches are then used to anchor the global alignment (Bray, Dubchak & Pachter 2002).

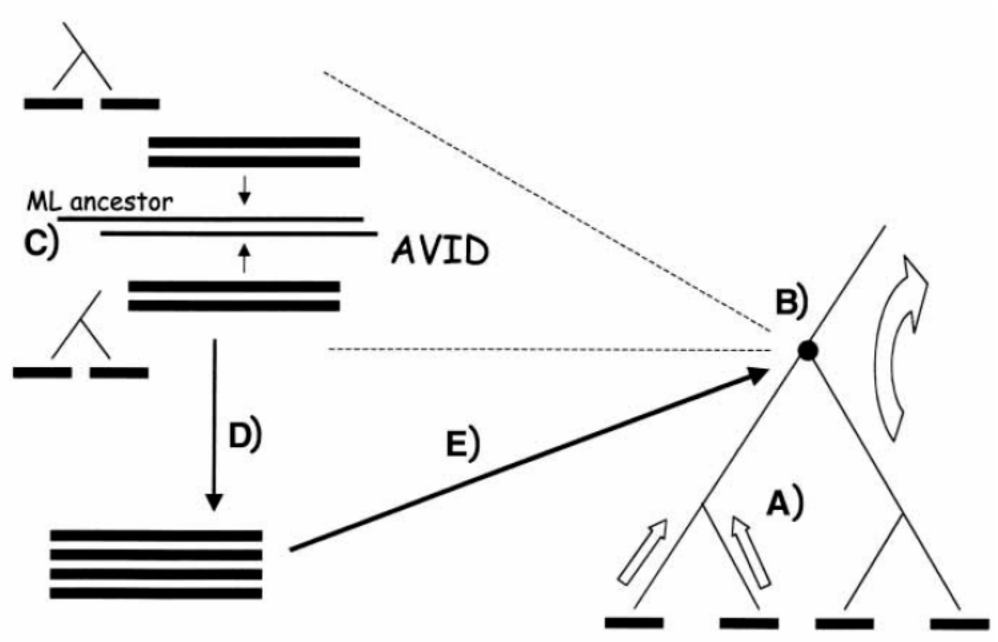

Figure 1 MAVID architecture overview. (*A*) Sequences are aligned upward along a guide tree and (*B*) alignments of alignments are performed at internal nodes. To align two alignments (*C*), maximum likelihood ancestor sequences are inferred from each of the separate alignments, and (*D*) the ancestor sequences are aligned with MAVID. The resulting multiple alignment (*E*) (corresponding to a subset of leaves of the tree) is then recorded at the internal node.

Figure 3 MAVID MSA procedure from (Bray & Pachter 2004)

4. Phylogenic Reconstruction

'This is a huge, complicated, and highly contentious field. However, always remember that regardless of algorithm used, parsimony, any distance method, maximum likelihood, or even Bayesian techniques, all molecular sequence phylogenetic inference programs make the absolute validity of your input alignment their first and most critical assumption. The accuracy of your alignment is the most important factor in inferring reliable phylogenies; the results are utterly dependent on its quality.'(Thompson, S. M. 2004, p. 11)

4.1 Introduction

Phylogeny is the field of biology that is concerned with identifying and understanding the relationship between species based on ancestor/descendant relationships. The phylogeny of organisms is usually represented by an evolutionary tree (or a cladogram).

Phylogeography is the study of the relationship between the phylogenic variations within or between species and their geographic distribution. Chloroplast DNA is particularly appropriate for phylogeographic studies since cpDNA does not recombine and inheritance is mostly maternal (Freeman et al. 2001).

Phylogenetic analysis of DNA or amino acid sequences involves four steps: sequence alignment, determination of the substitution model, tree building and tree evaluation (Baxevanis & Ouellette 1998).

As detailed in *Bioinformatics* (Baxevanis & Ouellette 1998, pp. 189-230) the alignment procedure is central to the resulting evolutionary tree constructed during phyologenetic analysis . An alignment produces 'sites' -which are aligned base positions- and these sites are, in effect, assumed to be genealogically homologous. Therefore, the alignment procedure is inextricably linked to the phylogenetic analysis.

The tree building criteria has an effect upon the alignment and substitution models applied. There are three tree building criteria used for calculating the phylogeny: distance matrix, maximum parsimony (MP) and maximum likelihood (ML).

> "Distance trees use pairwise divergence estimates of all sequences in the data to determine tree topology and branch lengths. Maximum parsimony finds the tree that explains with the fewest number of discrete steps all the base differences in a multiple sequence alignment. Maximum likelihood finds the topology and branch lengths that have the highest probability of producing the observed multiple sequence alignment." (Baxevanis & Ouellette 1998, p. 191)

For even moderate numbers of sequences being analysed there is an exponentially large number of possible trees for which the 'best' representation of the evolutionary relationship must be selected. As a result the optimal phylogenic tree(s) may only be found when examining generally fewer than 11 species. When the number of sequences being analysed exceeds 11 but is less than approximately 20, branch-and-bound methods can be used to find optimal phylogenic trees. For larger data sets most tree estimates are found using an uphill searching algorithm. Uphill searching heuristics are employed in the common phylogenic software packages PAUP (Swoffard 1998) and Phyllip (Felsenstein 1993) (Salter 2000).

4.2 Substitution Models

The first stage in deducing an evolutionary tree is the multiple sequence alignment (section 3). During and after the alignment of the sequences in question a substitution model is required to aid in estimating the divergence and hence evolutionary relationship between each sequence. A substitution model is a square matrix (4x4 in the case of nucleotide sequences) that reflects the 'cost' of substituting one base for another, the diagonal entries represent the 'cost' of having the same base in different sequences (which must be 0 for parsimonious methods). Weighted parsimony is when such a substitution matrix is fixed prior to computing a tree. In distance matrix and maximum likelihood tree building the costs are calculated from instantaneous rate matrices whereby the costs are a relative rate between the different substitutions and the frequency of the target base. (Baxevanis & Ouellette 1998, pp. 189-230)

There are numerous available substitution models as summarised in the table below (further details of the substitution models can be seen in appendix A).

Model	Summary	Assumptions
Number of Differences	Counts the number of sites at which two sequences being compared are different.	
Jukes-Cantor	Gives ML estimate of the number of substitutions between two sequences.	Equal substitution rates.
Kimura-2Parameter	Takes into account transitional and transversional substation rates.	Nucleotide frequencies are equal and rates of substitution are invariant.
p-distance	The proportion of nucleotide sites at which two sequences being compared are different. This model makes no correction for multiple substitutions or for rate biases (eg differences in transitional / transversional rates).	Evolutionary rate amongst sites constant and uniform rate bias.
Tajima-Nei	When nucleotide frequencies differ from 0.25 the Tajima-Nei distance can give better estimates of number of substitutions through weighting the substitution matrix based upon the observed frequencies.	Equality of substitution rates among sites and between transitional and transversional substitutions
Tamura-3Parameter	Takes into account differences in transitional and transversional rates and corrects for multiple hits	Equality of substitution rates amongst sites
Tamura-Nei	Corrects for multiple hits, takes into account differences in substitution rates between nucleotides and the inequality of nucleotide frequencies. It also distinguishes between transitional substitution rates between purines (nucleotides A or G) and transversional rates between pyrimidines (C or T).	Equality of substitution rates amongst sites

Table 4 Comparison of substitution models

4.3 Tree Building

Many tree building methods have been developed, including: branch-and-bound, branch swapping, star-decomposition, divide-and-conquer and stochastic search methods (such as simulated annealing, genetic algorithms or Markov Chain Monte Carlo methods). One of the most widespread tree building methods is a form of star decomposition called neighbour-joining (Saitou & Nei 1987) which uses a distance matrix criteria. This method has also been adapted for MP and ML criteria (Bruno, Socci & Halpern 2000; Yang 1997).

For n species or sequences under consideration there is:

$\frac{(2n-5)!}{|2^{n-3}(n-2)!|}$ possible unrooted trees, with an even greater number of possible trees for rooted trees. The methods employed to infer relationships and degree of divergence between sequences is beyond the scope of this study as it is a significant area of research in itself.

4.4 Tree Evaluation

Tree evaluation methods include: skewness test (randomised trees), permutation tests (randomised character data), resampling (bootstrapping, parametric bootstrapping, jackknife) and likelihood ratio tests. (Baxevanis & Ouellette 1998, pp. 213-7)

Felsenstein's bootstrap test (Felsenstein 1985) is one of the most widely used techniques to add reliability to an inferred phylogenic tree. Following a tree building algorithm (on *m* sequences of length *n*) from each sequence *n* nucleotides are randomly selected (with replacement) giving a new set of sequences. A tree is then reconstructed from this set of sequences using the same tree building method. The topology of the new tree is compared to the original tree, so that for each interior branch of the original tree that is different to the reconstructed tree the sequences it partitions are given a score of 0 whilst all other interior branches are given a score of 1. The process of reconstructing trees is repeated several hundred times, with the percentage of times an interior branch gains a value of 1 determining the level of bootstrap support for that branch. As a general rule an interior branch is considered “correct” if it receives a 95% or higher bootstrap value (Baxevanis & Ouellette 1998, pp. 221-2).

5. Bioinformatics Resources

"The computational biology community in general has been very generous with the fruits of its labour, making freely accessible tools and specialized databases without which elementary sequence analysis could not take place." (Baxevanis & Ouellette 1998)

Bioinformatics is a recent and fast moving field. There is currently a myriad of available software packages and online references. This chapter shall note some of the most commonly used packages, languages and websites for those looking at analysing DNA sequences.

5.1 Online Databases

The International Nucleotide Sequence Database Collaboration (INSDC) maintains the major DNA sequence databases. INSDC is comprised of the DNA Data Bank of Japan (*DDBJ Homology Search System*), the European Molecular Biology Laboratory Nucleotide Sequence Database and the NCBI operated Genbank (see table 5 below). These databases collaborate to share new submissions and as such are synonymous although they do use different formats (Vision & McLysaght 2004).

Database	**URL**
DDBJ	www.ddbj.nig.ac.jp
EMBL	www.ebi.ac.uk/embl/index.html
GenBank	www.ncbi.nlm.nih.gov/Genbank/index.html

Table 5 Major DNA databases and locations

There also exists databases holding protein sequences such as the Protein Information Resource (Wu et al. 2003) and Swiss-Prot, databases containing protein structure such as the Protein DataBank (Berman et al. 2000) as well as the Molecular Modeling DataBase at NCBI (Wang, Y. et al. 2002). The exponential growth in entries within these databases has been a motivating factor for efficient heuristic algorithms for sequence comparison.

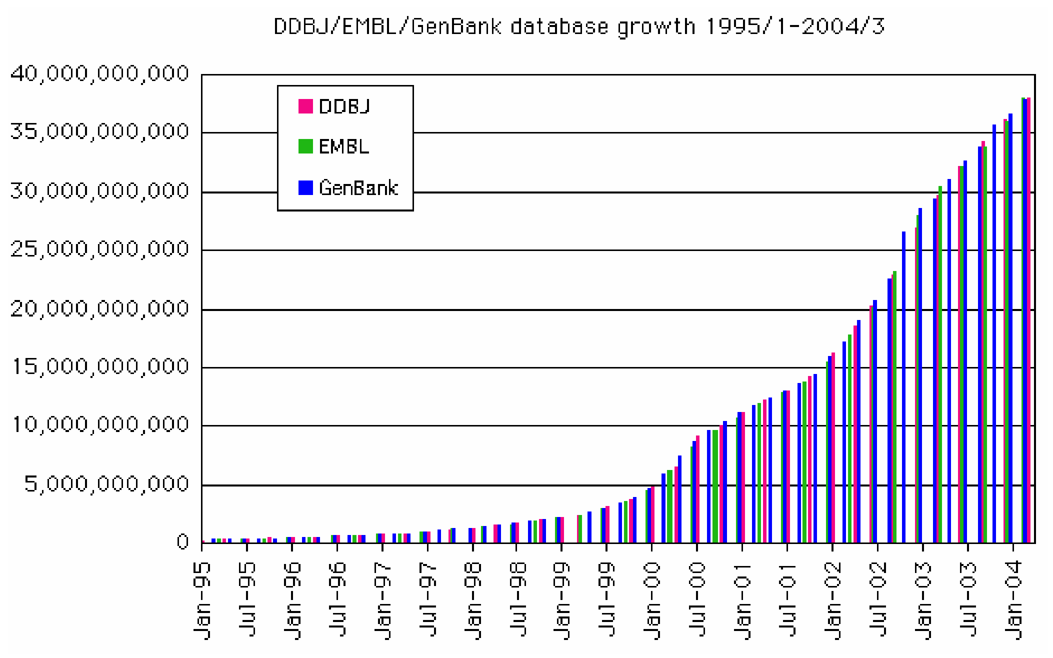

Figure 4 Growth of the nucleotide databases (number of nucleotides held) from January 1995 to January 2004

5.2 Software

Coupled with the databases listed above are many freely available online tools for searching the databases. One such collection of tools is ENTREZ which provides an entry point for many of the searching tools and resources provided by NCBI

Many of the tools used in analysing DNA and protein sequences are freely available and open source software. There are also programs available for conversion between sequence formats, for multiple sequence analysis and for phylogenic analysis (sections 5.2.1 and 5.2.2). A non-complete list of software available is included below.

5.2.1 Multiple Sequence Alignment

A list of the many algorithms and programs available for multiple sequence alignment problems are shown below:

Program	Algorithm	URL	Reference
MSA	Exact	ftp://fastlink.nih.gov/pub/msa/	(Lipman, Altschul & Kececioglu 1989)
DCA	Exact	http://bibiserve.techfak.uni-biefeld.de/dca	(Stoye, Moulton & Dress 1997)
OMA	Iterative DCA	http://bibiserve.techfak.uni-biefeld.de/oma	(Reiner, Stoye & Will 2000)
ClustalW	Progressive	ftp://ftp-igbmc.u-strasbg.fr/pub/clustalW	(Thompson, J., Higgins & Gibson 1994)
MultAlin	Progressive	http://ww.toulouse.inra.fr/multalin.html	(Corpet)
DiAlign	Consistency-based	http://www.gsf.de/biodv/dialign.html	(Morgenstern, Dress & Wener 1996)
ComAlign	Consistency-based	http://www.daimi.au.df/ocaprani	(Bucka-Lassen, Caprani & Hein 1999)
T-Coffee	Consistency-based, Progressive	http://igs-server.cnrs-mrs.fr/~cnotred	(Notredame, Holm & Higgins 1998)
IterAlign	Iterative	http://giotto.Stanford.edu/~luciano/iteralign.html	(Brocchieri & Karlin 1998)
SAM	Iterative/Stochastic/HMM	rph@cse.ucsc.edu	(Hughey & Krogh 1996)
HMMER	Iterative/Stochastic/HMM	http://hmmer.wustl.edu/	(Eddy 1995)
SAGA	Iterative/Stochastic/GA	http://igs-server.cnrs-mrs.fr/~cnotred/	(Notredam & Higgins 1996)
POA	Partial order alignment	http://www.bioinformatics.ucla.edu/poa	(Lee, Grasso & Mark 2002)
GA	Iterative/Stochastic/GA	czhang@watnow.uwaterloo.ca	(Zhang & Wong 1997)

Table 6 Multiple sequence alignment algorithms

5.2.2 Phylogenic Analysis

Some of the major phylogeny programs are shown below:

Application	Criteria	URL	Operating System
PHYLIP	MP & ML	http://evolution.genetics.washington.edu/phylip.html	Windows, Mac, Unix
PAUP		http://paup.csit.fsu.edu/	Mac, Unix, Dos, Windows
HYPHY	ML	http://www.hyphy.org/	Mac, Windows, Unix
PAML	ML	http://abacus.gene.ucl.ac.uk/software/paml.html	Windows, Unix, Mac OSX, Linux

Table 7 Phylogenic packages

5.3 Programming

There are currently many tools available to computer scientists and those with an interest in bioinformatics to program tailored solutions to particular or generic bioinformatics problems. The Open Bioinformatics Foundation is an organisation supporting open source programming in bioinformatics. The Open Bioinformatics Foundation (*O|B|F OPEN BIOINFORMATICS FOUNDATION*) contains projects in BioPerl, BioJava, BioPython, BioRuby, BioPipe, BioSQL / OBDA, Moby and DAS. These projects include freely available methods and scripts to handle standard tasks such as file conversion and database searching amongst others.

There is also the R project for statistical computing (Ihaka & Gentleman 1996). The bioconductor project is an open-source and open-development software project for the analysis of genomic data making use of the R program (*BioConductor*; Ihaka & Gentleman 1996).

6. Methodology

6.1 Aims

The J_{LA} and an extended J_{LA+} region of cpDNA of eucalypts has been found to be hypervariable (Freeman et al. 2001; Vaillancourt & Jackson 2000). The cpDNA sequences from these regions have been used to investigate the evolution and biogeography of eucalypts (Freeman et al. 2001; Whittock 2000). The high level of variation within the J_{LA+} region of *E. globulus* has led to difficulties in creating an unambiguous alignment of the sequences for phylogenetic analysis. As a consequence sequences have been aligned by eye using Sequence Navigator software (Karplus K & Hu 2001; Whittock 2000). The treatment of gaps in constructing a multiple sequence alignment on such variable sequences is problematic. For this reason an exact multiple sequence alignment algorithm was implemented and compared with new and existing heuristic MSA algorithms.

The alignment generated by the biologist was used as a benchmark alignment to compare the MSA generated with POA, clustalW, MAVID, DiAlign and AAE. Descriptive features of these alignments were analysed along with other statistics detailed in section 6.4. Phylogenic trees were then constructed from the alignments and analysed for differences, and support (see section 6.5).

The analysis of cpDNA samples also aimed to identify inversions and other rearrangement mutations within the data set and to determine if inversions (if any) were having a detrimental effect upon the multiple sequence alignment.

The data set of cpDNA was provided by Simon Whittock from the school of Plant Science. The analysis was conducted on the subgenus *Symphyomyrtus* data set (which displayed a maximum sequence divergence of 5%), and more specifically on the *maidenaria* subset which displayed a reduced sequence divergence of 3% (Whittock 2000).

6.2 Implementation

The raw data sequences obtained from plant science were of all samples of *Symphomytrus* (71 species). The sequences were already aligned in the nexus file format. This format is used by PAUP and other programs and aims to be an extensible multiple sequence format. The Nexus file format contains a header block with information regarding the number of species, the aligned length and information on the types of sequences (DNA/Amino Acid) the gap character and the character for any missing data.

A number of Perl scripts were written to parse out the individual sequences from this file and remove all gaps so that the raw sequence data was available for analysis. This was achieved through Perls extensive regular expression functionality. Perl scripts were also written to analyse the alignments generated by SLAGAN and to parse out all alignments which contained inversions (see code snippet below).

```
if ($gdata =~ m/\-/){
      print logf "found an inversion in: $filea\_$fileb \n";
      @nums = ($gdata =~ m/(\d+\.?\d*|\.\d+)/g);
      $x1 = $nums[1];
      $y1 = $nums[2];
      $x2 = $nums[8];
      $y2 = $nums[9];
      print logf "* $filea\_$fileb - between: \($x1 , $y1 ) \($x2 ,
$y2) ";
```

The Aligning Alignments Exactly (Kececioglu & Starrett 2004) algorithm was developed in C++ based on the grid structure and recurrences detailed in section 3.31. The AAE program requires its input to be: sequence(s) file A, sequence(s) file B, output filename, substitution matrix filename, “-dom” for dominance pruning speedup. The AAE program is able to read in a substitution matrix in the following format:

```
     A    C    G    T    -    N
A   91 -114  -31 -123    0  -43
C -114  100 -125  -31    0  -43
G  -31 -125  100 -114    0  -43
T -123  -31 -114   91    0  -43
-    0    0    0    0    0    0
N  -43  -43  -43  -43    0  -43

-750 -25
```

In the above substitution matrix (nucmatrix), substitutions between different DNA bases are prescribed different penalties, with matching DNA characters all scoring highly (either 100 or 91), the numbers in the final line of the matrix represent the gap open penalty (-750) and the gap extension penalty (-25). Most substitution matrices are developed for amino acid substitutions; with the exception of the above matrix as used in LAGAN (Brudno et al. 2003), the clustalW1.6 matrix and the IUB matrix (see Appendix A).

The AAE algorithm was implemented using modified sequence and multi-sequence objects developed by Brudno as a part of the LAGAN source code (licenced under GPL). The algorithm was implemented with the dominance pruning speedup and was capable of aligning all sequences in the *maidenaria* set. Descriptive features of the resultant alignment are shown in section 7.2 with the MSA shown in Appendix B.

6.3 Analysis of Inversions

To detect inversions in the sequences the shuffle-lagan (SLAGAN) algorithm (Brudno et al. 2003) was run in a pair-wise manner over all the sequences in the data set. From this algorithm a .mon file containing the co-ordinates and direction (+/-) of the chains that make up the best 1-monotonic conservation map is created for each pairwise alignment. The .mon files for these pairwise alignments were parsed using Perl's regular expressions to find any inversions (-) in the 1-monotonic conservation map. A list of the sequences for which inversions were found in the SLAGAN pairwise alignment was written to file.

The initial (21) pairwise alignments generated by clustalW based on the guide tree were examined to determine if inversions were present. The number and length of any inversions present give an indication of the extent to which inversions may hamper tradition MSA algorithms.

The results for detecting inversions is presented in section 7.1.

6.4 Analysis of Alignments for Phylogenic Reconstruction

Thompson et al. introduced two measures for comparing an alignment to a reference alignment: the sum of pairs score (SPS) and the column score (CS) (Thompson, J., Plewniak & Poch 1999). The SPS score increases as more sequences are correctly aligned and signifies the extent to which the algorithm succeeded in aligning the sequences in the alignment. The column score gives a measure of the ability of an algorithm to align all of the sequences correctly, however only one character in any column must be different causing a zero score. For this reason the CS was not used as a measure of how the generated alignments compared with the benchmark alignment

The SPS statistic used is the ratio of the all residue pairs that are aligned in the test alignment against the sum of all residue pairs in the reference alignment. That is, for an alignment A, of N sequences and M columns, and with column *i* from A represented as A_{i1}, A_{i2}, .. , A_{iN} . Then defining $p_{ijk} = 1$ if residues A_{ij} and A_{ik} are aligned. Then the score for the ith column S_i is given by:

$$S_i = \sum_{j=1}^{N} \sum_{\substack{j \neq k \\ k=1}}^{N} p_{ijk}$$

Then for a reference alignment R, of M_R columns and using S_{Ri} to denote the score for the ith column from alignment R, the SPS score is given by:

$$SPS = \sum_{i=1}^{M} S_i \div \sum_{i=1}^{M_R} S_{Ri}$$

This formula allows for a comparison of the two MSA and the degree to which they have successfully aligned all the sequences. The above formula has also been modified to score 2 for identically aligned pairs of residues, a 1 for aligned gaps and 0 otherwise (Karplus K & Hu 2001). This scoring has been called the weighted sum of pairs score (W-SPS).

Most MSA programs aim to maximise an objective function however when analysing constructed alignments against verified alignments (such as those in BaliBase test suite) often the weighted sum of pairs score is higher in the test alignments than the 'true' alignment (Lassman Timo & Erik 2002, p. 127).

In using a MSA for phylogenic analysis each column and the variations within it are considered, and as such, those columns which contain the same character over all sequences are uninformative. Maximum parsimony methods of constructing an evolutionary tree include only those sites which exhibit at least two different nucleotides, and for which those nucleotides occur at least twice, in its analysis. For this reason examining the percentage of parsimony informative sites over an entire MSA enables a comparison of the informative content of each MSA in creating an evolutionary tree using MP methods.

The percentage of singleton sites was scored. A singleton site is a site containing at least two types of nucleotides; with at most one occurring multiple times. Constant sites are sites for which there is only one nucleotide occurring over all the sequences. The percentage of constant sites was scored, along with the percentage of variable sites. A variable site is either parsimony informative or a singleton site and as such all sites are either variable or constant.

The number of gaps required to align the sequences is scored, along with the (aligned) length and the number of indels. An indels is an inserted gap greater than 1 base length, the total number of indels scored was the sum over all the sequences of the indels in each sequence. The p-distance of the alignments was also calculated. The p-distance is the proportion of sites for which two sequences being compared are different. The p-distance is calculated by dividing the number of differences found by the total number of nucleotides compared and is averaged by the number of sequences.

The following multiple sequence alignment programs were used in the comparison: POA, ClustalW, Aligning Alignments Exactly, DiAlign and MAVID along with a number of substitution matrices. A table of the descriptive features of these alignments is presented in section 7.2.2. The speed, space complexity and other advantages/disadvantages of the aforementioned algorithms are detailed in section 7.2.1.

6.5 Phylogenic Analysis

Phylogenic analysis was conducted using MEGA version 3.0 (Kumar, Tamura & Nei 2004) which provides a user-friendly interface to phylogenic analysis. MEGA contains a number of substitution models (p-distance, jukes-cantor etc) as well as a number of tree building methods (Distance based -UPGMA, Neighbour-Joining, Maximum Parsimony, Minimal Evolution-ML). Due to the number of species under consideration a distance based method of tree building was chosen (UPGMA), with 500 replicates for bootstrap support. Maximum parsimony methods were also used on the two most promising alignments. Similarities and variations, along with the level of support for each phylogeny are detailed in section 7.3.

7. Results and Discussion

7.1 Rearrangement Mutations

Due to the hypervariable nature of the chloroplast DNA samples it was decided to use the SLAGAN algorithm to detect inversions and determine if these inversions (if any) were having a detrimental impact on the MSA. Using SLAGAN it is also possible to analyse other rearrangement events such as translocations and duplications.

After running SLAGAN in a pairwise manner over all the sequences, a PERL script was used on the SLAGAN output file detailing the components in the highest scoring monotonic map to parse out and log any inversions. 89 inversions were found (out of a possible 1722).

The inversions found were all of minor length and by using the guide tree calculated by clustalW only two of the initial pairwise alignments displayed any inversions. The alignment of *ovata919* and *crenulata* displayed inversions (see figure 6), as well as the alignment of *globHJ13cg9* and *ovata924* (see figure 7). The inversion in the *globHJ13cg9* and *ovata924* alignment was surprising since they have aligned well in the clustalW alignment (see Appendix B) as well as other alignments. The inversion in *ovata919* and *crenulate* is less surprising since there is more divergence within the first 200 residues of the alignment. Overall, whilst inversions were found, they appear to be few, short in overall length, and amongst more distantly related sequences. For these reasons it does not appear that inversions are a problem in creating a good alignment of the *maidenaria* data set.

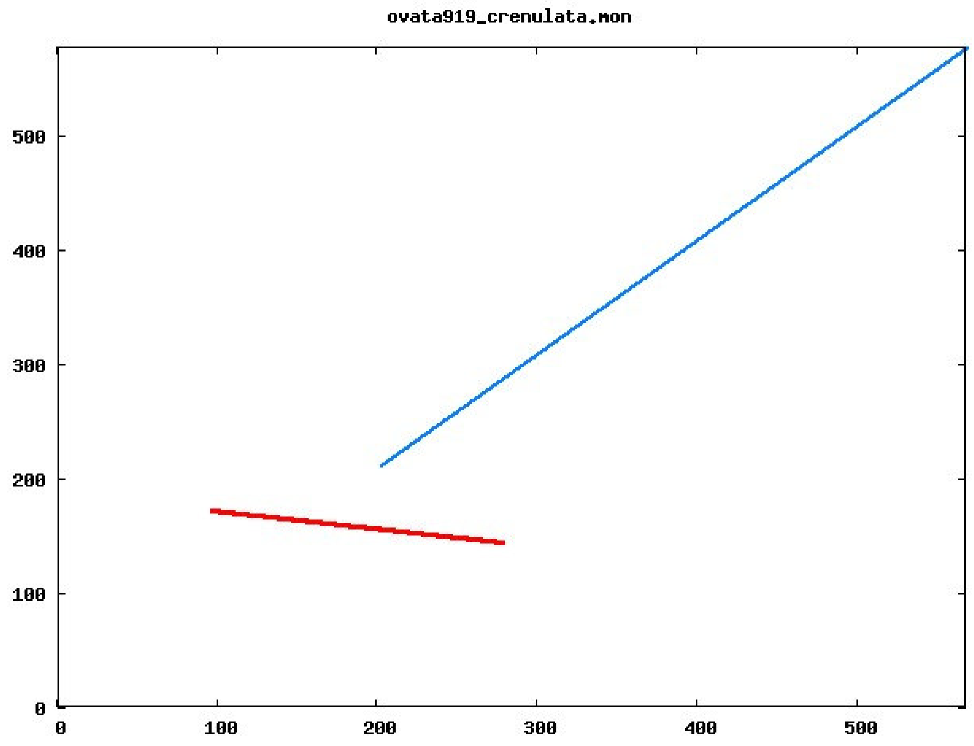

Figure 5 **Dot plot showing the inversion present in *ovata919* and *crenulata* alignment**

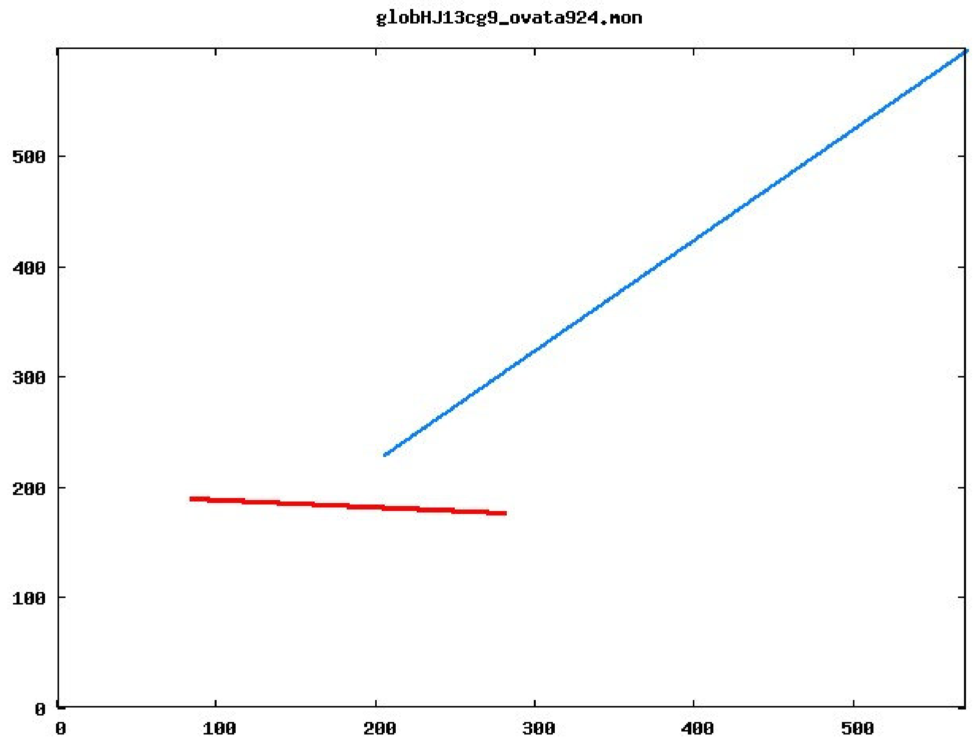

Figure 6 **Dot plot showing the inversion present in *globHJ13cg9* and *ovata924* alignment**

Translocation events were also found to be present in a number of the alignments as can be seen in figure 8.

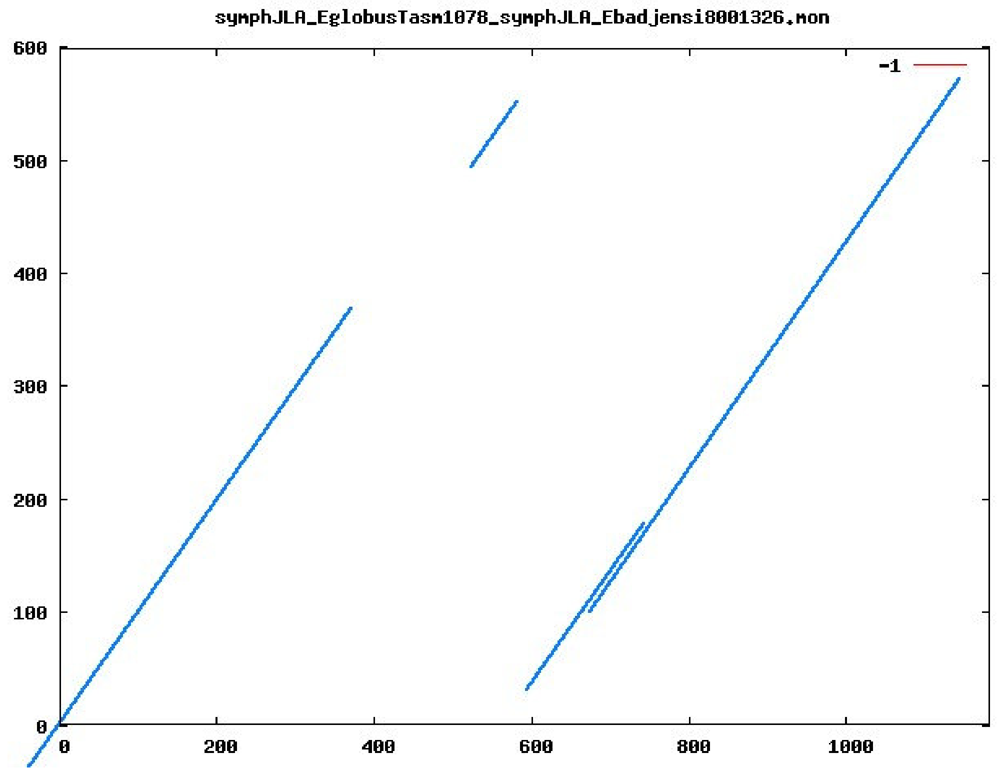

Figure 7 Example of translocation rearrangement in the pairwise alignment of *Eglobulus Tasmaniana1078* and *Ebadjensis8001326*

7.2 Multiple Sequence Alignment

7.2.1 Algorithmic complexity

The time and space complexity of the MSA alignments used are shown in the following table (table 8). POA is most suitable for MSA alignment problems over a large number of sequences since it does not require an initial guide tree calculation as used in clustalW and its efficient representation of alignments allows for greater speed.

Algorithm	**Background**	**Time Complexity**	**Space Complexity**
AAE (without speedups)	Alignments of *k* sequences of *n* columns. When aligning a sequence to a MSA the algorithm is polynomial.	Worst case: $\left\{\begin{matrix} \Theta\left(\left(3+\sqrt{2}\right)^{k}(n-k)^{2}k^{3/2}\right), k < n \\ \Theta\left(\left(3+\sqrt{2}\right)^{n}k^{2}n^{-1/2}\right), k \geq n \end{matrix}\right\}$	Worst case: factor of k better than time complexity
POA	Natural extension of dynamic programming for sequences of length N and M which is O(3NM). Depends upon number of predecessor nodes, if average number of predecessor nodes is n_p. Note n_p tends to increase slowly for biological meaningful alignments as poa is like a data compression algorithm	$O((2n_p+1)NM)$	O(N+L) where L is the total number of letters in all the sequences
ClustalW	ClustalW first performs pairwise alignments to construct guide tree before performing MSA routine. For N sequences of length L	$O(NL^2)$ to construct binary tree $O(L^2\log N)$ to construct MSA $O(NL^2+L^2\log N)$ overall	
MAVID	For N sequence of length L	linear in N and almost linear in L	
DiAlign	For N sequences of length L	$O(N^4L^2)$	

Table 8 Summary of the processing and memory complexity of MSA applications

7.2.2 MSA Features

The 42 sequences in the *maidenaria* subgenus were subjected to the MSA algorithms detailed in section 3.3. The 42 sequences in the *maidenaria* data set ranged in length between 463bp and 548bp (average length 522 bp). All alignments were generated using the default parameters.

The descriptive features of the alignments show that the benchmark alignment displayed the second lowest p-distance average, indicating that the biologists alignment is good in general. The MAVID alignment exhibited significantly more parsimony informative sites, whilst still retaining the most conserved sites overall and the lowest p-distance. These figures indicate that the alignment of all the sequences by MAVID produced a higher number of pairwise character matches, however this was at the expense of inserting more gaps leading to a longer alignment. This interpretation is confirmed by MAVID exhibiting the lowest W-SPS score (see table 10). The simplistic clustalW substitution matrix without dynamic GOP and GEP control was able to align the sequences over the shortest length (609bp) under the AAE algorithm, however this apparent strength is weakened by this alignment having a significantly higher p-distance score (see figure 8) along with SPS and W-SPS (see figure 11 below).

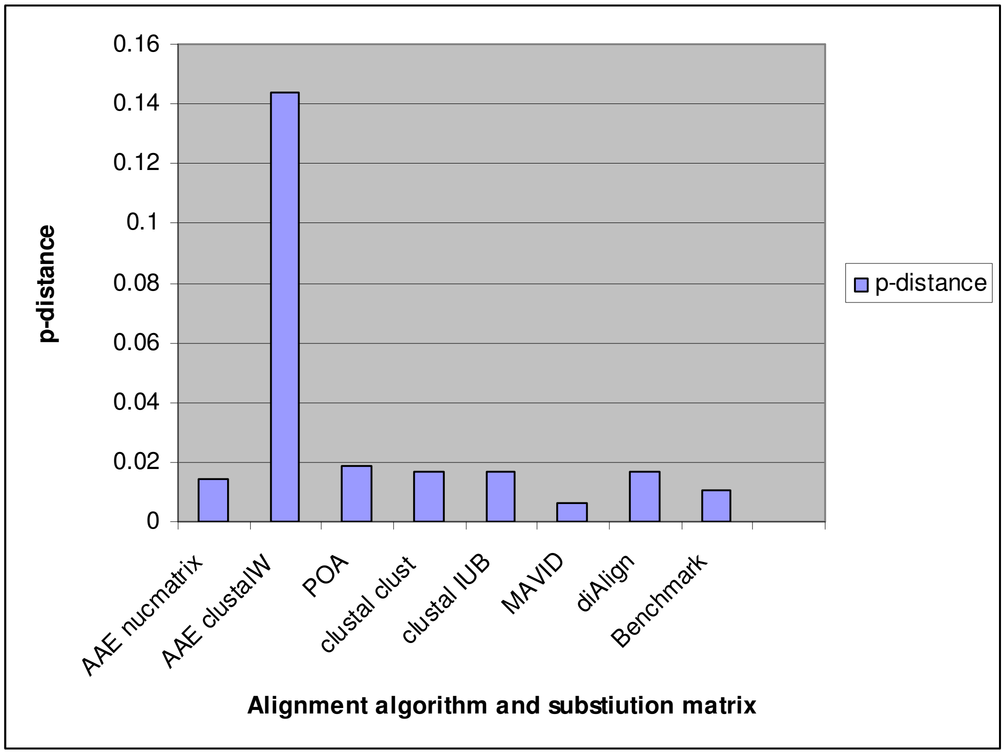

Figure 8 p-distance scores for all MSA

The exact algorithm (AAE) with the nucmatrix substitution matrix was able to align the *maidenaria* data set with significantly fewer indels as shown in figure 9. POA and MAVID aligned with the greatest numbers of indels (810 and 835 respectively) compared with the benchmark alignment which included 622 indels.

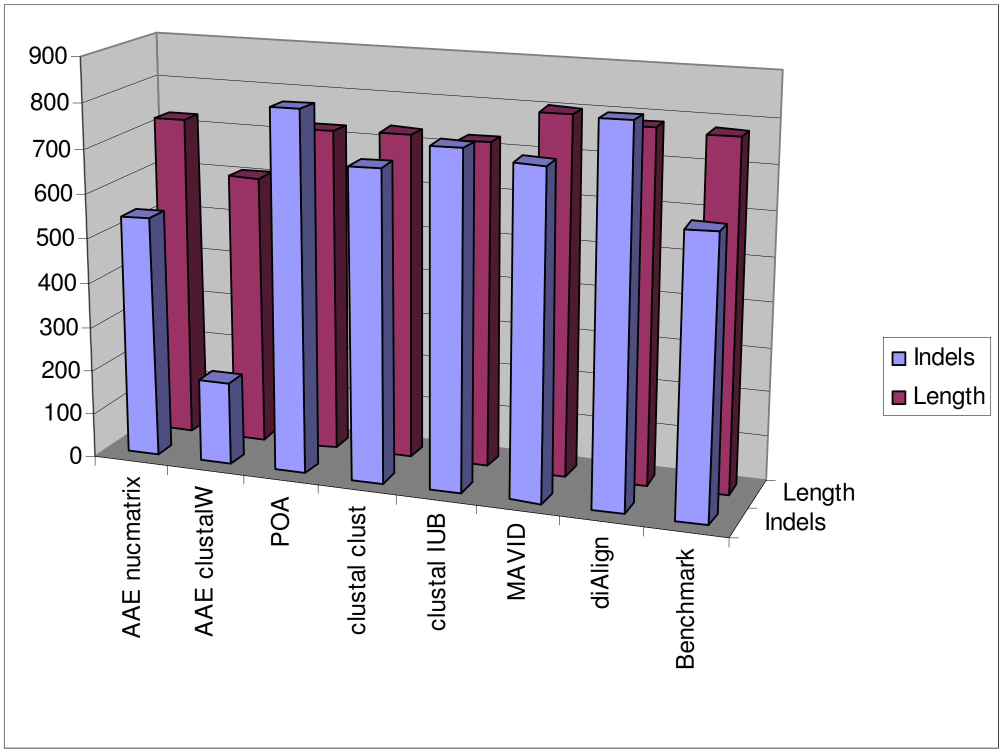

Figure 9 **MSA algorithms (with substitution matrix) of *maidenaria* data set showing aligned length (bp) and number of indels**

Both AAE with nucmatrix substitution matrix and POA were able to align the sequences over a far shorter length (with consequently fewer gaps inserted) and with a high number of conserved sites (76.13% and 76.92% respectively as compared with the benchmark alignment displaying 75.99%). Interestingly the benchmark alignment also scored the worst for parsimony informative sites (3.47% compared to 6.58% average over all other alignments). This apparent lack of informative sites was overcome through conversion of the alignment to a binary matrix listing characters as purine or pyrimidine.

Alignment Algo	Subst. Matrix	Length	Gaps	In-dels	CS %	PI S %	VS %	SS%	P-dist
POA	blosum80	728	9440	810	76.92	5.63	13.46	7.83	0.0187
MAVID	-	803	12893	730	83.31	7.10	15.44	8.09	0.0063
AAE	clustalW	609	3966	186	34.65	54.0	60.59	2.30	0.1441
AAE	nucmatrix	729	9486	543	76.13	6.72	15.50	7.68	0.0141
Clustal W	clustalW	734	9719	697	75.75	6.95	14.58	7.49	0.0171
Clustal W	IUB	728	9440	752	75.27	7.28	15.38	7.97	0.0170
DiAlign	-	786	11058	835	73.16	6.23	11.70	5.34	0.0171
Biologist	-	779	11786	622	75.99	3.47	8.60	5.13	0.0103

Table 9 Descriptive features of the aligned *maidenaria* data set.

Length is the aligned length of MSA, gaps are the total number of gaps inserted, indels is the number of insertion/deletion events over all sequences. CS% is the percentage of conserved sites, PI S% is the percentage of parsimony informative sites, VS% is the percentage of variable sites, SS% is the percentage of singleton sites.

The ability of the generated alignments to align residues over all the sequences compared with the benchmark biologist alignment is shown in table 10 as well as figure 10 below. The sum of pairs score and the weighted sum of pairs score give a measure of the quality of the generated alignments to the benchmark alignment. Whilst it has been noted that often the SPS or W-SPS score is greater than 1 (i.e. the test alignment contains a higher score than the 'good' reference alignment) , with the *maidenaria* data set this was never the case (Lassman Timo & Erik 2002).

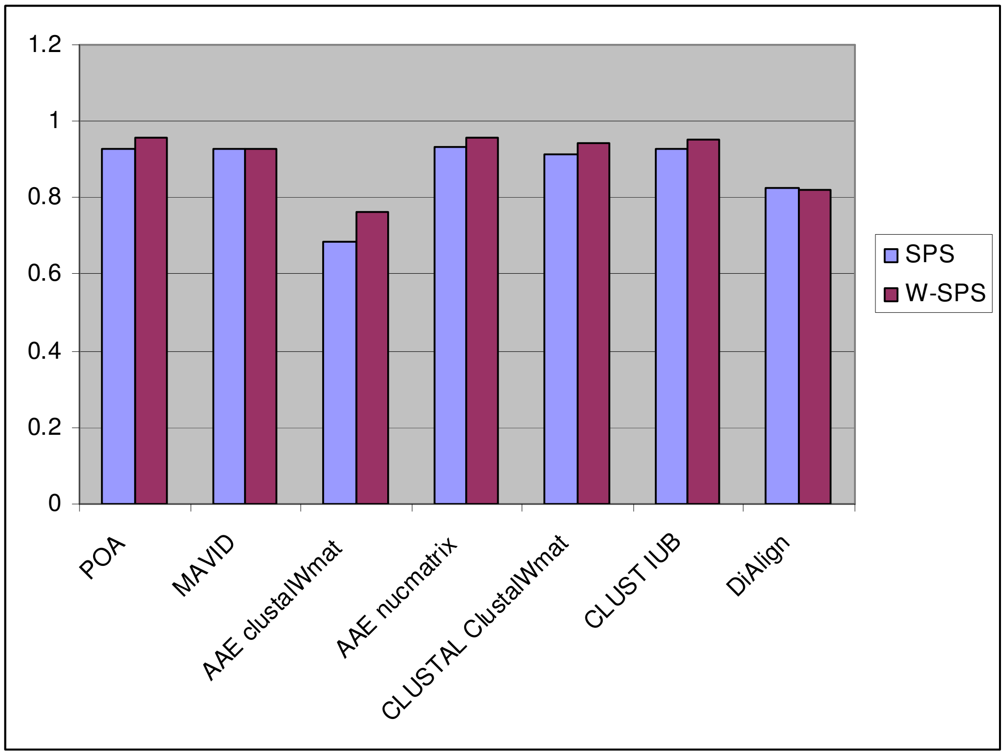

Figure 10 Sum of pairs (SPS) score and Weighted sum of pairs (W-SPS) for all MSA

All the algorithms achieved a close alignment (average 0.908 SPS) with the exception of DiAlign whose alignment was significantly poorer. The AAE algorithm with the nucmatrix achieved the highest SPS and weighted-SPS score over all algorithms, indicating that the alignment generated by this algorithm was able to produce an alignment that was overall very good compared to the benchmark alignment.

The SPS and W-SPS scores indicate that the local and global alignments are both able to achieve a high quality alignment on the *maidenaria* set of sequences, with MAVID (0.9297) scoring only marginally worse than the AAE (0.9305) algorithm.

Alignment Algo	**Subst Matrix**	**SPS**	**W-SPS**
POA	blosum80	0.9274	0.9547
MAVID	-	0.9297	0.9265
AAexact	ClustalW	0.6865	0.7606
AAexact	nucmatrix	0.9305	0.9570
ClustalW	clustalW	0.9138	0.9426
ClustalW	IUB	0.9261	0.9514
DiAlign	-	0.8249	0.8214

Table 10 MSA consensus statistics

7.3 Phylogenic Inference

Using the MSA generated through Aligning Alignments Exactly and the benchmark alignment phylogenic trees were generated using distance based. All trees were bootstrapped for 500 replications to add further support for the consensus tree generated.

The consensus tree generated through UPGMA distance based tree building method on the benchmark alignment is shown in figure 11. This tree is compared against the tree generated using the same method from the alignment constructed through AAE algorithm with the nucmatrix substitution matrix (figure 12).

The tree topology and distances are quite different in both trees, with *subcrenulata* sequence the most divergent in the benchmark tree with *globSA* showing as most divergent from the generated alignment. Most of the sequences showed similar groupings, however there was enough of a variation to warrant further investigation and analysis of the alignments and the variations within.

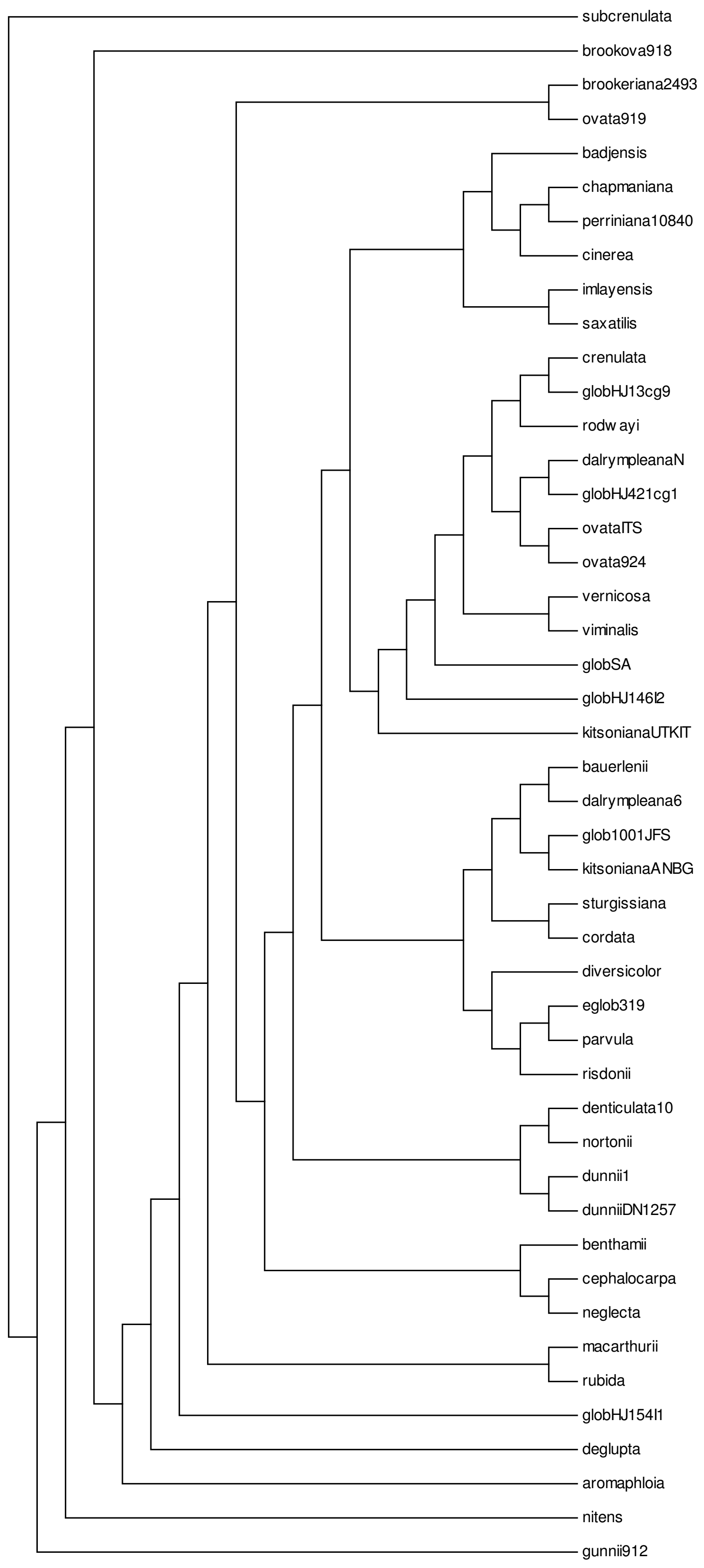

Figure 11 UPGMA consensus tree generated from benchmark alignment using p-distance substitution matrix

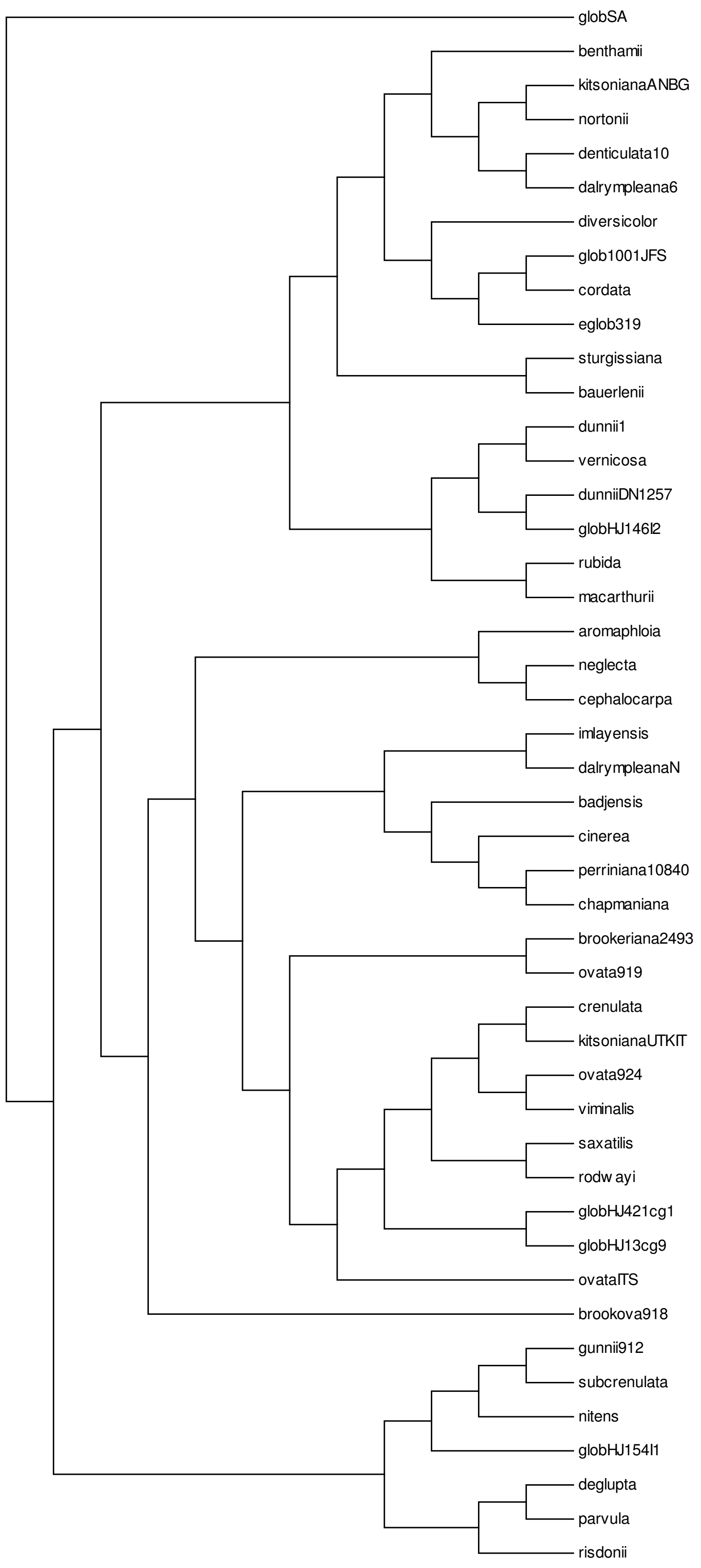

Figure 12 UPGMA consensus tree generated from alignment constructed using AAE with nucmatrix using the p-distance substitution matrix for tree building

8. Future Work

Much research continues to be undertaken in producing efficient and biologically relevant alignments. Research into incorporating dynamic parameter control, as used in ClustalW, as an option for partial order alignments and the aligning alignments exactly algorithms could allow for greater selectivity and specificity in aligning sequences.

Greater research and a review of the best performing algorithms on sequences exhibiting ranges of variation (length, divergence, motifs etc), along with parameter selection choices could be undertaken to allow for a better picture of algorithmic strengths as well as greater confidence for biologists and others in selecting the best of the many worthwhile algorithms for MSA. This research could allow for the development of an approach for pre-processing sequences allowing for a quick analysis of various rearrangement events and pertinent statistics (divergence, regions of high similarity) that could contribute information for anchoring based algorithms or to allow for better (or automated) algorithm and parameter selection.

Further work to improve the speed of the AAE algorithm to replicate those published would also allow for more efficient exact alignments, and allow biologists to apply this recent algorithmic advance.

9. Conclusion

The process of aligning multiple biological sequences for phylogenic inference is inherently difficult. As well as coupling two NP-complete endeavours (multiple sequence alignment and phylogenic tree building) the final product – an evolutionary tree – can only ever be correct based on the algorithmic and evolutionary assumptions made. This difficulty is compounded by the many competing choices for the alignment parameters, in choosing the alignment algorithm, in selecting the substitution model and finally in selecting a tree building criterion. It is a widespread practice for biologist to manually ammend an alignment to correct for the incorrect placement of gaps, often as a legacy of the progressive MSA algorithm.

This thesis has examined new and novel multiple sequence alignment algorithms against popular algorithms such as ClustalW. In examining the multiple sequence alignment algorithms their advantages and disadvantages both algorithmically and in terms of processing and space complexity has been studied as well as the suitability of the resulting alignments for further phylogenic analysis.

All analysis was conducted on 46 samples of chloroplast DNA of average length 522 bp taken from *eucalypts* provided by the School of Plant Science. The analysis of the multiple sequence alignments of these sequences showed only minor variations in the quality of the alignments generated. POA and AAE algorithms performed the best overall, producing alignments that were of minimal length, with good p-distance and a high SPS and W-SPS score.

10. References

Aho, A & Corasick, M 1975, 'Efficient string matching: an aid to bibliographic search', *Communications ACM*, vol. 18, pp. 333-40.

Baxevanis, A & Ouellette, BF 1998, *Bioinformatics A Practial Guide to the Analysis of Genes and Proteins*, 1 edn, John Wiley & Sons Inc.

Berman, H, Westbrook, J, Feng, Z, Gilliland, G, Bhat, T, Weissig, H, shindyalov, I & bourne, P 2000, 'The Protein Data Bank', *Nucleic Acids Research*, vol. 30, pp. 235-42.

BioConductor, viewed 7.5.2004 <http://www.bioconductor.org/>.

Blake, T *Chloroplast Genome Structure*, viewed 26.4.04 <http://hordeum.oscs.montana.edu/class/CHLORLEC.html>.

Bonizzoni, P & Vedova, GD 2001, 'The complexity of multiple sequence alignment with SP-score that is a metric', *Theoretical Computer Science*, vol. 259, no. 1, pp. 63-79.

Bray, N & Pachter, L 2004, 'MAVID: Constrained ancestral alignment of multiple sequences', *Genome Research*, vol. 14, pp. 693-9.

Bray, N, Dubchak, I & Pachter, L 2002, 'AVID: A Global Alignment Program', *Genome Research*, vol. 13, no. 1, pp. 97-102.

Brocchieri, L & Karlin, S 1998, 'Assymetric-iterated multiple alignments of protein sequences', *Journal of Molecular Biology*, vol. 276, pp. 249-64.

Brudno, M, Malde, S, Poliakov, A, Do, C, Couronne, O, Dubchak, I & Batzoglou, S 2003, 'Glocal Alignment: finding rearrangements during alignment', paper presented to Eleventh International Conference on Intelligent Systems for Molecular Biology, Brisbane, Australia.

Brudno, M, Do, C, Cooper, G, Kim, M, Davydov, E, Program, NCS, Green, E, Sidow, A & Batzoglou, S 2004, 'LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA', *Genome Research*, vol. 13, no. 4, pp. 721-31.

Bruno, W, Socci, N & Halpern, A 2000, 'Weighted Neighbor Joining: A Likelihood-Based Approach to Distance-Based Phylogeny Reconstruction', *Molecular Biology Evolution*, vol. 17, no. 1, pp. 189-97.

Bucka-Lassen, K, Caprani, O & Hein, J 1999, 'Combining many multiple alignments in one improved alignment', *Bioinformatics*, vol. 15, no. 2, pp. 122-30.

Chang, W & Lawler, E 1994, 'Sublinear approximate string matching and biological applications', *Algorithmica*, vol. 12, no. 4/5, pp. 327-44.

Chang, W & Marr, T 1994, 'Approximate string matching and local similarity', paper presented to 5th Annual Symposium on Combinatorial Pattern Matching.

Chantal, K & Gaston, G 1999, 'Near Optimal Multiple Sequence Alignment Using a Traveling Salesman Problem Approach', paper presented to Proceedings of the String Processing and Information Retrieval Symposium.

Corpet, F 1988, 'Multiple sequence alignment with hierarchical clustering', *Nucleic Acids Research*, vol. 16, pp. 10881-90.

D'Antonio, L 2003, 'Incorporating Bioinformatics in an algorithms course', paper presented to Innovation and Technology in Computer Science Education.

DDBJ Homology Search System, 2004, DDBJ, viewed 27.4.04 <http://spiral.genes.nig.ac.jp/homology/welcome-e.shtml>.

Eddy, S 1995, 'Multiple alignments using hidden Markov models', paper presented to Third international conference on intelligent systems for molecular biology, Cambridge, England.

Felsenstein, J 1985, 'Confidence limits on phylogenies: An approach using the bootstrap', *Evolution*, vol. 39, pp. 783-91.

---- 1993, *Phylogenetic Inference Package (PHYLIP)*, 3.5 edn, University of Washington.

Feng, D & Doolittle, R 1996, 'Progressive alignment of amino acid sequences and construction of phylogenic trees from them', *Methods Enzymol.*, vol. 266, pp. 368-82.

Freeman, JS, Jackson, HD, Steane, DA, McKinnon, GE, Dutkowski, GW, Potts, BM & Vaillancourt, RE 2001, 'Chloroplast DNA phylogeography of Eucalyptus globulus', *Australian Journal of Botany*, vol. 49, pp. 585-96.

Galil, Z & Park, K 1990, 'An improved algorithm for approximate string matching', *SIAM Journal of Computing*, vol. 19, no. 6, pp. 989-99.

Gibson, G & Muse, S 2002, *A Primer of Genome Science*, Sinauer Associates, Inc.

Gusfield, D 1999, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press.

Hughey, R & Krogh, A 1996, 'Hidden Markov models for sequence analysis: extension and analysis of the basic method', *Computer Applications in Biological Science*, vol. 12, pp. 95-107.

Ihaka, R & Gentleman, R 1996, 'R: A language for Data Analysis and Graphics', *Journal of Computational and Graphical Statistics*, vol. 5, no. 3, pp. 299-314.

Jukes, TH & Cantor, CR 1969, 'Evolution of protein molecules', in HN Munro (ed.), *Mammalian Protein Metabolism*, Academic Press, New York, pp. 21-132.

Junior, SAdC 2003, 'Sequence Alignment Algorithms', M.Sc. in Advanced Computing thesis, Kings College University of London.

Karplus K & Hu, B 2001, 'Evaluation of protein multiple alignments by SAM-T99 using BaliBASE multiple alignment test set', *Bioinformatics*, vol. 17, no. 8, pp. 713-20.

Kececioglu, J & Starrett, D 2004, 'Aligning Alignments Exactly', paper presented to RECOMB, San Diego, California, USA.

Kumar, S, Tamura, K & Nei, M 2004, 'MEGA3: Integrated software for Molecular Evolutionary Genetics Analysis and sequence alignment', *Bioinformatics*, vol. 5, no. 2 (In Press).

Lassman Timo & Erik, S 2002, 'Quality assesment of multiple alignment programs', *FEBS Letters*, vol. 529, pp. 126-30.

Lee, C, Grasso, C & Mark, S 2002, 'Multiple sequence alignment using partial order graphs', *Bioinformatics*, vol. 18, no. 3, pp. 452-64.

Lipman, D, Altschul, S & Kececioglu, J 1989, 'A tool for multiple sequence alignment', *Proc. Natl. Acad. Sci.*, vol. 86, pp. 4412-5.

Lopez, R *2Can Bioinformatics Educational Resource*, viewed 29/04/2004 <http://www.ebi.ac.uk/2can/home.html>.

Maizel, J & Lenk, R 1981, 'Enhanced graphic matrix analysis of nucleic acid and protein sequences', *Proc Natl Acad Sci USA*, vol. 78, p. 7665.

Morgenstern, B 1999, 'DIALIGN2: improvement of the segment-to-segment approach to multiple sequence alignment', *Bioinformatics*, vol. 15, no. 3, pp. 211-8.

Morgenstern, B, Dress, A & Wener, T 1996, 'Multiple DNA and protein sequence based on segment-to-segment comparison', *Proc. Natl. Acad. Sci.*, vol. 93, pp. 12098-103.

Muckstein, U, Hofacker, I & Stadler, P 2002, 'Stochastic pairwise alignments', *Bioinformatics*, vol. 18, no. Supplement 2, pp. S153-S60.

Navarro, G 2001, 'A Guided tour to approximate string matching', *ACM Computing Surveys*, vol. 33, no. 1, pp. 31-88.

Needleman, S & Wunsch, C 1970, 'A general method applicable to the search for similarities in the amino acid sequences of two proteins', *Journal of Molecular Biology*, vol. 48, pp. 444-53.

Notredam, C & Higgins, D 1996, 'SAGA: sequence alignment by genetic algorithm', *Nucleic Acids Research*, vol. 24, pp. 1515-24.

Notredame, C 2002, 'Recent progress in multiple sequence alignment: a survey', *Pharmacogenomics*, vol. 3, no. 1, pp. 131-44.

Notredame, C, Holm, L & Higgins, D 1998, 'COFFEE: an objective function for multiple sequence alignments', *Bioinformatics*, vol. 14, no. 5, pp. 407-22.

O|B|F OPEN BIOINFORMATICS FOUNDATION, viewed 7.5.2004 <http://open-bio.org/>.

Reiner, K, Stoye, J & Will, T 2000, 'An iterative method for faster sum-of-pair multiple sequence alignment', *Bioinformatics*, vol. 16, no. 9, pp. 808-14.

Saitou, N & Nei, M 1987, 'The neighbor-joining method: a new method for reconstructing phylogenetic trees', *Molecular Biology Evolution*, vol. 4, pp. 406-25.

Salter, L 2000, 'Algorithms for Phylogenetic Tree Reconstruction', paper presented to International Conference on Mathematics and Engineering Techniques in Medicine and Biological Science.

Schaal, B, Hayworth, D, Olsen, K, Rauscher, J & Smith, W 1998, 'Phylogeographic studies in plants: problems and prospects', *Molecular Ecology*, vol. 7, pp. 465-74.

Sellers, P 1974, 'On the theory and computation of evolutionary distances', *SIAM Journal of Applied Mathematics*, vol. 26, pp. 787-93.

---- 1980, 'The theory and computation of evolutionary distances: pattern recognition', *Algorithmica*, vol. 1, pp. 359-73.

Stoye, J, Moulton, V & Dress, A 1997, 'DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment', *Computational Applied Bioscience*, vol. 13, no. 6, pp. 625-6.

Swoffard, D 1998, *PAUP* Phylogenetic analysis using parsimony (*and other methods)*, 4 edn, Sinauer Associates.

Thompson, J, Higgins, D & Gibson, T 1994, 'CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting. position specific gap penalties and weight matrix choice', *Nucleic Acids Research*, vol. 22, no. 22, pp. 4673-80.

Thompson, J, Plewniak, F & Poch, O 1999, 'A comprehensive comparison of multiple sequence alignment programs', *Nucleic Acids Research*, vol. 27, no. 13, pp. 2682-90.

Thompson, SM 2004, 'Multiple Sequence Alignment and Analysis: A two part introduction and practical guide to the art and science of comparative genomics.' in RP Grant (ed.), *Computational Genomics: Theory and Applications*, Horizon Scientific Press, Norfolk, UK.

Ukkonen, E 1985, 'Finding approximate patterns in strings', *Journal of Algorithms*, vol. 6, pp. 132-7.

Vaillancourt, R & Jackson, H 2000, 'A chloroplast DNA hypervariable region in eucalypts', *Theoretical and Applied Genetics*, vol. 101, no. 5, pp. 473-7.

Vinga, S & Almeida, J 2003, 'Alignment-free sequence comparison- a review', *Bioinformatics*, vol. 19, no. 4, pp. 513-23.

Vision, T & McLysaght, A 2004, 'Computational Tools and Resources n Plant Genome Informatics', in P Christou & H Klee (eds), *Handbook of Plant Biotechnology*, 1 edn, John Wiley & Sons., p. 1552.

Wang, L & Jiang, T 1994, 'On the complexity of multiple sequence alignment', *Journal of Computational Biology*, vol. 1, no. 4, pp. 337-48.

Wang, Y, Anderson, J, Chen, J, geer, L, He, S, Hurwitz, D, Liebert, C, Madej, T, Marchler, G, Marchler-Bauer, A, Panchenko, A, Shoemaker, B, Song, J, Thiessen, P, Yamashita, R & Bryant, S 2002, 'MMDB: Entrez's 3D structure database', *Nucleic Acids Research*, vol. 30, pp. 249-52.

Whittock, S 2000, 'Phylogenetic Reconstruction with Maximum Likelihood', Reading Thesis thesis, University of Tasmania.

Wu, C, Yeh, L-S, Huang, H, Arminski, L, Castro-Alvear, J, Chen, Y, Hu, Z-Z, Ledley, R, Kourtesis, P, Suzek, B, Vinayaka, C, Zhang, J & Barker, WC 2003, 'The Protein Information Resource', *Nucleic Acids Research*, vol. 31, pp. 345-7.

Yang, Z 1997, *Phylogenetic analysis by Maximum Likelihood (PAML)*, 1.3 edn, Dept. of Integrative biology, University of California, Berkeley.

Zhang, C & Wong, A 1997, 'A genetic algorithm for multiple molecular sequence alignment', *Computational Applied Bioscience*, vol. 13, no. 6, pp. 565-81.

11. Appendices

A. Substitution Matrices

Kimura 2-parameter

	A	T	C	G
A	–	β	α	α
T	β	–	α	β
C	β	α	–	β
G	α	β	β	–

Tajima-Nei

	A	T	C	G
A	–	αG_T	αG_C	αG_G
T	αG_A	–	αG_C	αG_G
C	αG_A	αG_T	–	αG_G
G	αG_A	αG_T	αG_C	–

where G_A, G_T, G_C, G_G are the respective frequencies of A,T,C and G respectively.

Tamura 3-Parameter

	A	T	C	G
A	–	$\beta(1-\theta)$	$\beta\theta$	$\alpha\theta$
T	$\beta(1-\theta)$	–	$\alpha\theta$	$\beta\theta$
C	$\beta(1-\theta)$	$\alpha(1-\theta)$	–	$\beta\theta$
G	$\alpha(1-\theta)$	$\beta(1-\theta)$	$\beta\theta$	–

Tamura-Nei

	A	T	C	G
A	–	βG_T	βG_C	$\alpha_1 G_G$
T	βG_A	–	$\alpha_2 G_C$	βG_G
C	βG_A	$\alpha_2 G_T$	–	βG_G
G	$\alpha_1 G_A$	βG_T	βG_C	–

clustalW1.6 substitution matrix

	A	C	G	T	–	N
A	1	0	0	0	0	0
C	0	1	0	0	0	0
G	0	0	1	0	0	0
T	0	0	0	1	0	0

```
-    0    0    0    0    0    0
N    0    0    0    0    0    0

-15 -6.6
```

IUB

```
     A    C    G    T    -    N
A    1.9  0    0    0    0    1.9
C    0    1.9  0    0    0    1.9
G    0    0    1.9  0    0    1.9
T    0    0    0    1.9  0    1.9
-    0    0    0    0    0    0
N    1.9  1.9  1.9  1.9  0    1.9

-15 -6.6
```

Nucmatrix

```
     A    C    G    T    -    N
A   91 -114  -31 -123    0  -43
C -114  100 -125  -31    0  -43
G  -31 -125  100 -114    0  -43
T -123  -31 -114   91    0  -43
-    0    0    0    0    0    0
N  -43  -43  -43  -43    0  -43

-750 -25
```

Blosum80

```
#  Blosum80
#  Matrix made by matblas from blosum80.iij
#  * column uses minimum score
#  BLOSUM Clustered Scoring Matrix in 1/3 Bit Units
#  Blocks Database = /data/blocks_5.0/blocks.dat
#  Cluster Percentage: >= 80
#  Entropy =   0.9868, Expected =  -0.7442
GAP-PENALTIES=12 6 6
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X  ?  a  g  t  c  u  ]  n
A  7 -3 -3 -3 -1 -2 -2  0 -3 -3 -3 -1 -2 -4 -1  2  0 -5 -4 -1 -3 -2 -1 -9 -9 -9 -9 -9 -9 -9 -9
R -3  9 -1 -3 -6  1 -1 -4  0 -5 -4  3 -3 -5 -3 -2 -2 -5 -4 -4 -2  0 -2 -9 -9 -9 -9 -9 -9 -9 -9
N -3 -1  9  2 -5  0 -1 -1  1 -6 -6  0 -4 -6 -4  1  0 -7 -4 -5  5 -1 -2 -9 -9 -9 -9 -9 -9 -9 -9
D -3 -3  2 10 -7 -1  2 -3 -2 -7 -7 -2 -6 -6 -3 -1 -2 -8 -6 -6  6  1 -3 -9 -9 -9 -9 -9 -9 -9 -9
C -1 -6 -5 -7 13 -5 -7 -6 -7 -2 -3 -6 -3 -4 -6 -2 -2 -5 -5 -2 -6 -7 -4 -9 -9 -9 -9 -9 -9 -9 -9
Q -2  1  0 -1 -5  9  3 -4  1 -5 -4  2 -1 -5 -3 -1 -1 -4 -3 -4 -1  5 -2 -9 -9 -9 -9 -9 -9 -9 -9
E -2 -1 -1  2 -7  3  8 -4  0 -6 -6  1 -4 -6 -2 -1 -2 -6 -5 -4  1  6 -2 -9 -9 -9 -9 -9 -9 -9 -9
G  0 -4 -1 -3 -6 -4 -4  9 -4 -7 -7 -3 -5 -6 -5 -1 -3 -6 -6 -6 -2 -4 -3 -9 -9 -9 -9 -9 -9 -9 -9
H -3  0  1 -2 -7  1  0 -4 12 -6 -5 -1 -4 -2 -4 -2 -3 -4  3 -5 -1  0 -2 -9 -9 -9 -9 -9 -9 -9 -9
I -3 -5 -6 -7 -2 -5 -6 -7 -6  7  2 -5  2 -1 -5 -4 -2 -5 -3  4 -6 -6 -2 -9 -9 -9 -9 -9 -9 -9 -9
L -3 -4 -6 -7 -3 -4 -6 -7 -5  2  6 -4  3  0 -5 -4 -3 -4 -2  1 -7 -5 -2 -9 -9 -9 -9 -9 -9 -9 -9
K -1  3  0 -2 -6  2  1 -3 -1 -5 -4  8 -3 -5 -2 -1 -1 -6 -4 -4 -1  1 -2 -9 -9 -9 -9 -9 -9 -9 -9
M -2 -3 -4 -6 -3 -1 -4 -5 -4  2  3 -3  9  0 -4 -3 -1 -3 -3  1 -5 -3 -2 -9 -9 -9 -9 -9 -9 -9 -9
F -4 -5 -6 -6 -4 -5 -6 -6 -2 -1  0 -5  0 10 -6 -4 -4  0  4 -2 -6 -6 -3 -9 -9 -9 -9 -9 -9 -9 -9
P -1 -3 -4 -3 -6 -3 -2 -5 -4 -5 -5 -2 -4 -6 12 -2 -3 -7 -6 -4 -4 -2 -3 -9 -9 -9 -9 -9 -9 -9 -9
S  2 -2  1 -1 -2 -1 -1 -1 -2 -4 -4 -1 -3 -4 -2  7  2 -6 -3 -3  0 -1 -1 -9 -9 -9 -9 -9 -9 -9 -9
T  0 -2  0 -2 -2 -1 -2 -3 -3 -2 -3 -1 -1 -4 -3  2  8 -5 -3  0 -1 -2 -1 -9 -9 -9 -9 -9 -9 -9 -9
W -5 -5 -7 -8 -5 -4 -6 -6 -4 -5 -4 -6 -3  0 -7 -6 -5 16  3 -5 -8 -5 -5 -9 -9 -9 -9 -9 -9 -9 -9
Y -4 -4 -4 -6 -5 -3 -5 -6  3 -3 -2 -4 -3  4 -6 -3 -3  3 11 -3 -5 -4 -3 -9 -9 -9 -9 -9 -9 -9 -9
V -1 -4 -5 -6 -2 -4 -4 -6 -5  4  1 -4  1 -2 -4 -3  0 -5 -3  7 -6 -4 -2 -9 -9 -9 -9 -9 -9 -9 -9
B -3 -2  5  6 -6 -1  1 -2 -1 -6 -7 -1 -5 -6 -4  0 -1 -8 -5 -6  6  0 -3 -9 -9 -9 -9 -9 -9 -9 -9
Z -2  0 -1  1 -7  5  6 -4  0 -6 -5  1 -3 -6 -2 -1 -2 -5 -4 -4  0  6 -1 -9 -9 -9 -9 -9 -9 -9 -9
X -1 -2 -2 -3 -4 -2 -2 -3 -2 -2 -2 -2 -2 -3 -3 -1 -1 -5 -3 -2 -3 -1 -2 -9 -9 -9 -9 -9 -9 -9 -9
? -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9
a -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9  4 -2 -2 -2 -2 -9  0
g -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -2  4 -2 -2 -2 -9  0
t -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -2 -2  4 -2  4 -9  0
c -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -2 -2 -2  4 -2 -9  0
u -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -2 -2  4 -2  4 -9  0
] -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9
n -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9 -9  0  0  0  0  0 -9  0
```

B. Multiple Sequence Alignments

See attached CD

11. Appendices