<table>
<thead>
<tr>
<th><strong>Title</strong></th>
<th>The impact of guide trees in large-scale protein multiple sequence alignments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Authors(s)</strong></td>
<td>Boyce, Kieran</td>
</tr>
<tr>
<td><strong>Publication date</strong></td>
<td>2016</td>
</tr>
<tr>
<td><strong>Publication information</strong></td>
<td>Boyce, Kieran. “The Impact of Guide Trees in Large-Scale Protein Multiple Sequence Alignments.” University College Dublin. School of Medicine, 2016.</td>
</tr>
<tr>
<td><strong>Publisher</strong></td>
<td>University College Dublin. School of Medicine</td>
</tr>
<tr>
<td><strong>Item record/more information</strong></td>
<td><a href="http://hdl.handle.net/10197/8698">http://hdl.handle.net/10197/8698</a></td>
</tr>
</tbody>
</table>
The Impact of Guide Trees in Large-Scale Protein Multiple Sequence Alignments

Kieran Boyce
12255781

This thesis is submitted to University College Dublin in fulfilment of the requirements for the degree of Doctor of Philosophy

School of Medicine
Head of School: Prof. Patrick Murray

Principal Supervisor: Prof. Desmond G. Higgins

Doctoral Studies Panel: Dr. Peadar O’Gaora (chair)
Dr. Fabian Sievers
Prof. Desmond G. Higgins

September 2016
# Contents

1 Introduction .......................... 1
  1.1 Overview .......................... 1
  1.2 Multiple Sequence Alignments ....... 4
    1.2.1 Objective function .......... 5
    1.2.2 Optimisation strategy ..... 8
  1.3 Guide trees ........................ 11
    1.3.1 Guide trees versus phylogenetic trees .. 12
    1.3.2 Similarity calculations ..... 14
    1.3.3 Clustering .................. 15
    1.3.4 Iterations .................. 17
  1.4 Benchmarking ........................ 19
    1.4.1 Consistency among alignment methods .. 20
    1.4.2 Simulated sequences .......... 21
    1.4.3 Phylogenetic tests .......... 22
    1.4.4 Structural benchmarks ..... 23
  1.5 Large alignments .................. 27

2 Chained Guide Trees ................. 29
  2.1 Introduction ........................ 29
  2.2 Methods .......................... 31
  2.3 Results .......................... 34
    2.3.1 Initial analysis ............ 34
    2.3.2 BAiBASE .................... 36
    2.3.3 Larger datasets ............. 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.4 Small models</td>
<td>42</td>
</tr>
<tr>
<td>2.3.5 HomFam</td>
<td>43</td>
</tr>
<tr>
<td>2.3.6 Optimal ordering</td>
<td>44</td>
</tr>
<tr>
<td>2.4 Complexity and run times</td>
<td>48</td>
</tr>
<tr>
<td>2.5 Chaining in other situations</td>
<td>48</td>
</tr>
<tr>
<td>2.6 Subsequent work</td>
<td>50</td>
</tr>
<tr>
<td>2.7 Discussion</td>
<td>55</td>
</tr>
<tr>
<td>3 Instability in Alignment Algorithms</td>
<td>58</td>
</tr>
<tr>
<td>3.1 Introduction</td>
<td>58</td>
</tr>
<tr>
<td>3.2 Methods</td>
<td>60</td>
</tr>
<tr>
<td>3.2.1 HomFam</td>
<td>60</td>
</tr>
<tr>
<td>3.2.2 Software</td>
<td>61</td>
</tr>
<tr>
<td>3.2.3 Supporting material</td>
<td>64</td>
</tr>
<tr>
<td>3.3 Results</td>
<td>64</td>
</tr>
<tr>
<td>3.3.1 Alignment instability</td>
<td>64</td>
</tr>
<tr>
<td>3.3.2 Unique distances</td>
<td>66</td>
</tr>
<tr>
<td>3.3.3 Distance measure calculations</td>
<td>67</td>
</tr>
<tr>
<td>3.3.4 Theoretical maximum number of unique distances and sequences</td>
<td>73</td>
</tr>
<tr>
<td>3.3.5 Smaller alignments</td>
<td>74</td>
</tr>
<tr>
<td>3.3.6 Algorithm symmetry</td>
<td>77</td>
</tr>
<tr>
<td>3.4 Conflicting instability</td>
<td>77</td>
</tr>
<tr>
<td>3.5 Discussion</td>
<td>78</td>
</tr>
<tr>
<td>4 Dynamic Ordering of Sequences</td>
<td>81</td>
</tr>
<tr>
<td>4.1 Introduction</td>
<td>81</td>
</tr>
<tr>
<td>4.2 Methods</td>
<td>83</td>
</tr>
<tr>
<td>4.2.1 HomFam</td>
<td>83</td>
</tr>
<tr>
<td>4.2.2 Algorithm</td>
<td>84</td>
</tr>
<tr>
<td>4.2.3 Alignment programs</td>
<td>87</td>
</tr>
<tr>
<td>4.2.4 Supporting material</td>
<td>89</td>
</tr>
<tr>
<td>4.3 Results</td>
<td>89</td>
</tr>
</tbody>
</table>
4.3.1 Static versus dynamic ordering ........................................ 89
4.3.2 Balibase ................................................................. 90
4.3.3 HomFam ................................................................. 93
4.3.4 Instability ............................................................... 97
4.3.5 An alternative quality measure ....................................... 101
4.4 Run-Time Performance .................................................. 102
4.5 Discussion ................................................................. 108

5 Summary and Future Work ................................................. 111

Bibliography ........................................................................... 116

Appendices ............................................................................ 131

A Data files, Utility programs and Scripts ............................... 132
Abstract

The focus of this thesis is on large-scale progressive protein multiple sequence alignment algorithms. Although first developed over 30 years ago, multiple sequence alignment algorithms are still an active area of research given their widespread use in many biological analyses, and the dramatic increase in sequence information over the years. The behaviour of the existing algorithms with large numbers of sequences is examined in this work, and in particular the impact of guide trees on the alignments generated.

This thesis is divided into 5 chapters. Chapter 1 introduces the concept of a multiple sequence alignment, its uses and how it is constructed. It also details the specifics of progressive alignments, describes how guide trees are constructed, and provides an overview of a number of the ways in which the quality of an alignment can be measured.

Chapter 2 examines the impact the topology of the guide tree has on the generated alignment. It finds that simply aligning sequences one after another can produce higher quality alignments than the default alignment methods when measured using structure-based benchmarks. This increase in quality is particularly noticeable with larger alignments. It also finds that randomly ordering the sequences produces alignments with similar quality as any of the other orderings examined.

Chapter 3 finds that, because of a tradeoff between alignment accuracy and computation time, larger alignments generated by some of the most common multiple sequence alignment programs are inherently unstable, and changing the order in which the sequences are listed in the input file will cause a different alignment to be created.

Chapter 4 proposes an ordering of the sequences to be aligned that will produce a better quality alignment than the random ordering identified in Chapter 2. It also attempts to resolve the instability issue identified in the previous chapter.

Finally, Chapter 5 reviews the findings presented in the thesis, and proposes possible future steps to both use and continue to develop these findings.
Declaration

I hereby certify that the submitted work is my own work, was completed while registered as a candidate for the degree stated on the Title Page, and I have not obtained a degree elsewhere on the research presented in this submitted work.

Kieran Boyce
September 2016
Acknowledgement

Firstly, I would like to express my gratitude to my supervisor Prof. Desmond G. Higgins for his continuous support during my studies, for his patience, profound knowledge, sense of humour, and for giving me a chance. Sincere thanks also to Dr. Peadar O’Gaora and Dr. Fabian Sievers of my Doctoral Studies Panel, for their encouragement, insightful comments, and answering a myriad of the most rudimentary questions with unending patience.

I thank my fellow colleagues within the sequence alignment lab, Dr. Quan Le, Gearoid Fox and Peter Jehl, and current and past colleagues in adjacent labs and throughout the Conway Institute, Dr. Thomas Schwarzl, Dr. Graham Hughes, Dr. Markus Schroeder, Dr. Kate Killick, Paul Donovan, Gael Jalowicki, Lisa Rogers, Ines Freitas and Paul Lavin. Thank you for the companionship on the journey, and for making it so enjoyable.

My studies and this thesis would not have been possible without funding from Science Foundation Ireland. Thank you for giving me the opportunity to do this.

Special thanks to my mother and late father, who gave me a thirst for learning, and who always said I had the handwriting of a doctor. It would have been so much easier to learn how to write properly.

Finally, thank you Gráinne for putting up with my chasing another dream, for listening to my struggles, for the sacrifices over the last number of years, and for always being my light in the darkness.
Chapter 1

Introduction

1.1 Overview

The creation of a multiple sequence alignment is a fundamental step, often the first, in the comparative analysis of homologous genes or proteins. A sequence alignment is rarely the end product of an analysis exercise, but is further processed in a wide variety of workflows or pipelines. The most common applications of sequence alignments have been phylogenetic tree construction (Löytynoja and Goldman, 2005; Ogden and Rosenberg, 2006; Liu et al., 2010a), protein structure prediction (Marks et al., 2012; Jones et al., 2012; Taylor et al., 2012) and function prediction (Watson et al., 2005; Brandt et al., 2010; Radivojac et al., 2013). However, with the growth of sequence information and enhanced computational approaches, multiple sequence alignments now form the basis for numerous other applications, many of which are described in Kemena and Notredame (2009), Thompson et al. (2011) and Chatzou et al. (2015).

A multiple sequence alignment is an arrangement of biological sequences so that similar sequence features are aligned or arranged together. The purpose of this arrangement is to reveal patterns that may be shared by many sequences, or to highlight modifications that may explain functional or phenotypic variability. Typically the sequences are laid out in a rectangular array, one sequence per row, with the goal that residues in a particular column are either homologous (derived from one position in a common ancestral sequence), superposable (have the same local 3D structure), or have an equivalent functional role (Edgar and Batzoglou, 2006), depending on the purpose of the alignment. While these criteria are essentially the same for closely-related sequences, they may give different alignments when used with more distantly-related sequences.

Most multiple sequence alignment applications work with protein sequences, and in some sit-
valuations these are back-translated into nucleic acid sequences if necessary. Genome alignment programs have a number of unique challenges, specifically the extreme length of the genome, poor conservation, rearrangements, and gain, loss or duplication of large segments (Darling et al., 2010). Similarly, RNA sequence alignment programs have their own specific concerns and they usually combine both the primary sequence and secondary structure when generating the alignment (Nawrocki et al., 2009). However, the relationship between sequence and structure conservation drops off considerably for alignments with less than 60% sequence identity (Gardner et al., 2005), and structure tends to be more conserved than sequence (Capriotti and Marti-Renom, 2010). When structure is conserved, structurally-adjacent residues covary, but most alignment programs ignore such information and assume each site is independent. Given their specific requirements, neither DNA nor RNA sequence alignment programs will be examined in any detail in this thesis.

The most widely used approach for aligning multiple protein sequences is what is termed ‘progressive alignment’ by Feng and Doolittle (1987). The method was originally described by Hogeweg and Hesper (1984), and was also described by Barton and Sternberg (1987), Taylor (1988) and Higgins and Sharp (1988). In this, a similarity measure is first calculated between each pair of sequences to be aligned. A ‘guide tree’ (Higgins et al., 1992) is then created by clustering the sequences based on the similarity measures into a dendogram, and the sequences are then aligned into larger and larger alignments based on the branching order of the guide tree. While this is a heuristic approach that is not guaranteed to find the optimal alignment of the given sequences, it does allow alignments of large numbers of sequences to be quickly made on what is, by most reasonable definitions, computers with very modest processing power (Higgins and Sharp, 1988).

Progressive alignment has enabled the widespread use of multiple sequence alignments, as evidenced by the range of applications listed above and the inclusion of the article describing the alignment program Clustal W (Thompson et al., 1994) as the tenth most cited scientific paper of all time (Van Noorden et al., 2014). Algorithm development in multiple sequence alignment still remains an active area of research, and a number of challenges still remain to be solved, one of which is the scalability of the existing methods in light of the massive amounts of protein sequence data now available. Given that some of the most commonly used algorithms were initially developed 30 years ago, their appropriateness for use with such large datasets should be examined.

The focus of this thesis is on large-scale progressive protein multiple sequence alignment algorithms, and in particular the impact of guide trees on the alignments generated. For clarity, large-
scale in the context of this thesis is in excess of about one thousand sequences. With the existing state-of-the-art, large alignments result in low quality scores, as measured by structure-based benchmarks, and may take a considerable amount of time to produce. In particular, the alignment quality scales poorly with increasing numbers of sequences to be aligned. Historically, multiple sequence alignments were optimised for their sequence similarity, as a protein’s primary sequence was the only data available for the vast majority of proteins. While a number of alignment programs incorporate additional information e.g. including structural information (Barton and Sternberg, 1987; O’Sullivan et al., 2004) or using secondary structure predictions (Wright, 2015), this thesis will only examine alignment algorithms that use a protein’s primary sequence.

This thesis is divided into 5 chapters:

• In Chapter 2 Chained Guide Trees, the impact of the guide tree topology on alignment quality is examined. The main finding is that simply aligning sequences one after another, even if the sequences are randomly ordered, can result in higher quality alignments when measured using structure-based benchmarks. In addition, this approach greatly simplifies the construction of the guide tree to be used to order the sequence alignment, a step which is usually the performance bottleneck in alignment programs. Both the increase in quality and reduction in runtimes is particularly beneficial for larger alignments.

• Chapter 3 Instability in Alignment Algorithms examines one aspect of the tradeoff between alignment accuracy and computation time and finds that, because of a loss of information in the early steps of the progressive alignment approach, larger alignments generated by some of the most common multiple sequence alignment programs are inherently unstable, and changing the order in which the sequences are listed in the input file will cause a different alignment to be created.

• In Chapter 4 Dynamic Ordering of Sequences, an attempt is made to resolve two of the findings of the previous chapters. Specifically is there a better ordering of sequences than random when using chained guide trees, and can the instability in alignment algorithms be removed so that an alignment is not dependent on the order in which the sequences are read by the alignment program?

• Finally, in Chapter 5 Summary and Future Work, the findings presented in the thesis are reviewed, work by other researchers building on these findings are briefly described, and
possible future steps to both use and continue to develop these findings are outlined.

The remainder of this chapter provides greater detail on multiple sequence alignments and the rationale for a heuristic approach, the different means of determining alignment quality, and different ways of creating and using guide trees.

1.2 Multiple Sequence Alignments

Given a set of protein sequences, the goal of a multiple sequence alignment is to arrange the sequences in such a way that the evolutionary, structural or functional relationship between the individual sequences becomes apparent. This is achieved by inserting gaps of varying length in different positions in the sequences so that, in the resulting alignment, the residues in each column maximise their similarity according to the criteria in question. Hence, if the goal is to determine evolutionary relationships between proteins, residues should be homologous and the gaps represent insertions or deletions (indels) that may have occurred in the proteins’ evolution from their last common ancestor. Figure 1.1 shows an alignment of 6 sequences from the BAliBASE (Thompson et al., 2005) BB12021 reference set using Clustal Omega (Sievers et al., 2011) with the aim of identifying sequence homology.

![Figure 1.1: Alignment of the BAliBASE BB12021 reference set using Clustal Omega. Below the alignment itself, the three histograms are: Conservation — a measure of the number of physicochemical properties conserved for each column (Livingstone and Barton, 1993); Quality — the likelihood of observing mutations (if any) in a particular column, based on the relevant BLOSUM62 (Henikoff and Henikoff, 1992) scores; and Consensus — the percentage of the modal residue per column (‘+’ indicates more than one residue with the modal value). Output produced by Jalview (Waterhouse et al., 2009).](image-url)
1.2.1 Objective function

Multiple sequence alignment is essentially an optimisation exercise. Hence, the scoring function or objective function that is being optimised is critically important and must be able to discriminate between different possible alignments in terms of the relationship under investigation. As Notredame et al. (1998) point out, scoring functions tend to fall into one of two categories. The first of these rely on a substitution matrix that provide a score for each possible amino acid substitution, and gap penalty values that define the cost of insertions and deletions. These substitution matrix scores are derived from a probabilistic model and give a measure of the relative likelihood of a residue pair occurring as an aligned pair as opposed to an unaligned pair. Dayhoff and Schwartz (1978) defined the PAM substitution matrices by obtaining substitution frequencies from alignments of very similar proteins, and then extrapolating this information to allow for longer evolutionary distances. The base matrix is termed 1 PAM, where PAM is an acronym for point accepted mutation and gives the probability of one amino acid being replaced by another in 100 residues. For a substitution matrix appropriate to longer evolutionary times, the matrix is raised to a power $n$ to represent an evolutionary interval $n$ times longer. PAM250 has been one of the most widely-used substitution matrices. However, Gonnet et al. (1992) point out that this extrapolation from short time interval substitutions does not reflect the relationship between more distantly related proteins, as the former are dominated by single base changes in codon triplets whereas all types of codon change were observed with increasing PAM distance. Since then, substitution matrices have been constructed from alignments of more distantly related proteins. One such set is the BLOSUM matrix set (Henikoff and Henikoff, 1992), created from the BLOCKS database (Henikoff and Henikoff, 1991). BLOSUM62 has become a standard for ungapped sequence matching, and BLOSUM50 for alignments with gaps. The suffix number indicates the percentage of identical residues used to cluster the sequences in the BLOCKS database when calculating the substitution frequencies, and hence lower numerical values are more appropriate for longer evolutionary time or more un-related sequences.

The alignment process then aims to maximise the sum of the substitution scores for each aligned pair of residues, minus the penalty values for gaps that are introduced. This calculation is known as the SP or sum-of-pairs score (Edgar and Batzoglou, 2006), and while it gives a mathematically optimal solution it may not necessarily be the biologically correct one. One issue is that the alignments produced often depend on the specific parameter values, particularly gap penalties, used by the method (Vingron and Argos, 1991). These methods also assign the same penalties for substitu-
tions, insertions and deletions occurring anywhere in a sequence, even though a set of sequences is likely to possess both reasonably conserved and highly variable regions (Krogh et al., 1994). Finally, as shown above, the substitution matrices and gap penalties used with such methods are usually derived from statistical analysis of large numbers of sequences, which may not be representative of the sequences that are being aligned.

Notredame et al.’s second type of scoring function aims to overcome these limitations. Hidden Markov Models (Rabiner and Juang, 1986) use position-specific scoring schemes that typically have been determined from an alignment of sequences in the same protein family. A Hidden Markov Model (HMM) is a probabilistic model that describes a series of observations emitted by a “hidden” stochastic or Markov process. The HMM is made up of a number of states, which in this context correspond to the columns in a multiple sequence alignment (Eddy, 1996). Each state emits a symbol (residue) according to a set of emission probabilities, and the different states are connected by state-transition probabilities. As the emission probabilities are specific to a particular state or residue, HMMs can use conserved patterns and position-dependent gap penalties that can either be learned from the sequences being aligned or from an alignment of similar sequences generated previously.

HMMs were widely used in speech recognition since the 1970s (see Rabiner (1989) for historical details), and Churchill (1989) was one of the first to introduce them to computational biology. HMMs allow the application of formal probabilistic methods to a wide variety of tasks, including multiple sequence alignments (Eddy et al., 1995). Krogh et al. (1994) constructed an HMM profile from a training set of unaligned sequences (although they did not use that term), and then used it to create a multiple alignment of the training sequences. Eddy (1997) defines a HMM profile as a linear model of the consensus primary structure of a sequence family, including the position-specific scores for both amino acids and insertions/deletions at each profile position. Gribskov et al. (1987) first introduced the concept of a profile, and several other groups were working on similar approaches at that time — “flexible patterns” (Barton, 1990) and “templates” (Taylor, 1986). An example of an HMM and corresponding profile are given in Figure 1.2.

In general there are three common problems that HMMs address (Rabiner, 1989):

- Given an existing HMM and an observed sequence, what is the probability that the HMM could generate the sequence? Also known as the evaluation problem, it can be interpreted as scoring how well a given model matches a given observed sequence. A typical usage would
Figure 1.2: Top: A tiny example alignment of four sequences with three aligned “consensus” columns (upper case amino acid codes), and one insertion relative to the consensus (lower case amino acid code). Bottom: A cartoon view of an HMM-profile built from the same alignment. An HMM is composed of three things: states, state transition probabilities, and symbol emission probabilities. HMM “states” (circles) each align to one residue (or no residue). Match states (labeled M) align to a particular residue with some “emission probability” (illustrated schematically above each state, with a black bar for each residue of size proportional to the probability of that residue). Emission probabilities are usually calculated from the observed counts of residues in the corresponding column of the alignment. Delete (D) and insert (I) states allow for deletions and insertions relative to the consensus. A product of state transition probabilities (arrows) defines the probability of any given path (alignment) through the HMM. Reproduced from Eddy (1997, p3).
be for searching for homologous proteins in a database.

- What is the optimal state sequence that the HMM would use to generate the observed sequence? While there are typically many state sequences that could generate the observed sequence, the Viterbi algorithm (Viterbi, 1967) will find the most probable state path given the data. This can be used to align an existing sequence or profile to the HMM.

- Given sufficient data, what are the structure and parameters of the HMM which best account for the data? This is the training problem, and while there is no optimal way of estimating the model parameters, the Baum-Welsh method (Baum et al., 1970) or expectation-maximisation (EM) method (Dempster et al., 1977) can find a local optimum iteratively.

HMMs have a number of advantages over other scoring schemes, aside from the position-specific scores. The parameters of an HMM can be determined from unaligned as well as aligned data (Eddy, 1996). In addition, the treatment of insertions and deletions has a justifiable statistical basis, rather than the gap penalties being determined by trial and error. As the treatment of insertions and deletions is a particular problem with highly divergent protein sequences, HMMs are able to distinguish between members and non-members of a protein family with a high degree of accuracy (Krogh et al., 1994). However, since HMMs are built on the premise of a Markov process, they do not deal well with correlations between residues as they assume that the residue emitted depends only on one underlying state (Eddy, 2004). Hence, they are usually inappropriate for secondary structure analysis where long-range pairwise correlations are known to exist (Jones et al., 2012).

1.2.2 Optimisation strategy

Regardless of the scoring function used, the optimisation problem itself remains challenging. The best alignment of two sequences can be determined quite readily, using the algorithms described by Needleman and Wunsch (1970) for the alignment of the full length of the two sequences (global alignment), or Smith and Waterman (1981) for identifying subsequences sharing high similarity (local alignment), both of which are based on Bellman’s dynamic programming approach (Bellman, 1953). Unfortunately Wang and Jiang (1994) has shown that multiple sequence alignment with SP score is NP-complete, and Durbin et al. (1998) give the memory complexity as $O(\bar{L}^N)$ and the time complexity as $O(2^N \bar{L}^N)$, where $N$ is the number of sequences being aligned and $\bar{L}$ their average
length. Carrillo and Lipman (1988) have shown that the volume of the multidimensional dynamic programming matrix can be reduced, and this algorithm was implemented in the alignment program MSA (Lipman et al., 1989). However, even with this reduction in complexity, it is still only feasible to align a very small number of sequences using SP-score methods.

For aligning anything more than a few tens of sequences, the only feasible approach is to rely on approximate or heuristic algorithms to generate the multiple sequence alignment. By far the most common of these approaches is progressive alignment, which was initially described by Hogeweg and Hesper (1984). This involves including sequences into the overall alignment based on the branching order specified by a guide tree. The guide tree itself is computed based on the similarity of the sequences to be aligned, and the alignment order is then determined by traversing the tree from the leaves to the root. At each node either a pair of sequences, a sequence and a profile or a pair of profiles are aligned until the overall alignment is generated once the guide tree root is reached. As noted earlier, while this approach does not guarantee an optimal alignment, it is fast and simple and produces reasonable alignments. Some of the most commonly used alignment programs, particularly for creating large alignments, use this approach. Clustal Omega (Sievers et al., 2011) converts sequences and intermediate alignments into HMMs and aligns these using HHalign (Söding, 2005). Muscle (Edgar, 2004b) uses a measure called the log-expectation score in a highly optimised profile-to-profile alignment method. The Mafft (Katoh et al., 2002) FFT-NS algorithms use Fast Fourier Transforms for very fast pairwise alignments. Kalign (Lassmann et al., 2009) uses the Myers and Miller (1988) space-saving dynamic programming approach. While the alignment step may vary, in all of these programs the alignment order is specified by the guide tree that is created based on the similarity of the sequences to be aligned.

The primary drawback with progressive alignment is that it is a myopic or greedy algorithm, and that only the information available in the sequences or profiles currently being aligned is taken into consideration, rather than all the information available in the full set of sequences. As a result, while each alignment step may result in a good alignment at that point in the process, it may turn out that such alignments are not optimal in the context of the overall alignment. Hence, mistakes made early in the process are never corrected and can lead to poor alignments.

A number of other approaches have been proposed to overcome the greedy nature of progressive alignment. Some of these involve aligning all the sequences simultaneously as the MSA alignment program does, or use stochastic heuristics such as simulated annealing (Ishikawa et al.,
1993a; Kim et al., 1994) or genetic algorithms (Ishikawa et al., 1993b; Notredame and Higgins, 1996) rather than trying to calculate the precise optimum value of the SP score. While they do not guarantee an optimal alignment, these approaches have lower time and memory complexity, and therefore don’t have the same restrictions on either the numbers or lengths of sequences that can be aligned. Also, as Notredame et al. (1998) points out, these heuristic approaches can use different objective functions. As a result, they are not restricted to the information content of the sequences in that alignment step but can use the additional information available from the other sequences which should lead to a better overall alignment. There is a balance to be struck in how much additional information to include — too little and the algorithm is greedy but too much and the best overall alignment may be drowned out in noise.

One approach that has been proposed to find such a balance is termed consistency, invented by Notredame et al. (2000). Murata et al. (1985) note that when more than two sequences are aligned, pairwise comparisons rarely lead to a consistent alignment of the sequences. Gotoh (1990) proposed that consistent regions across all pairwise alignments would serve as “anchor points” which would then direct the alignment of the remaining “inconsistent” regions. DIALIGN (Morgenstern et al., 1998) finds ungapped local alignments using segment-to-segment comparisons rather than by comparing single residues. It assigns weights based on a probabilistic function, and then finds a consistent set of segment matches that maximises the sum of weights. As the authors point out in an earlier work (Morgenstern et al., 1996), consistency here means that the overall order of the positions in each sequence is respected. They also avoid the issue of determining a suitable gap penalty by not considering gaps in the calculation of the alignment score.

In T-Coffee, Notredame et al. (2000) use a library of all-against-all pairwise alignments from the set of sequences to be aligned, which can be built from a range of data sources. It then uses a progressive approach to generate the overall alignment, but it considers the alignments between all the pairs in the library at each alignment step. Probcons (Do et al., 2005) also uses a consistency-based approach, in that HMM-derived posterior probabilities are calculated for each pair of sequences and that this information is then included in the sum-of-pairs scoring function used during the progressive alignment. MSAProbs (Liu et al., 2010b) also calculates all pairwise posterior probability matrices using a pair-HMM, but incorporates sequence weighting in the progressive alignment. Finally, Mafft algorithms G-INS-i and L-INS-i (Katoh et al., 2005) also include all pairwise alignment information into the objective function. While the incorporation of this additional information does
give more accurate alignments, particularly for divergent sequences (Edgar and Batzoglou, 2006),
this improvement comes with a considerable performance cost (Sievers et al., 2011, 2013). Hence
consistency-based alignment programs are typically only used for aligning under 1,000 sequences.

One final aspect of the progressive alignment approach should be noted, namely the asymmet-
ric treatment of insertions and deletions. Löytynoja and Goldman (2005) point out that insertion
and deletion events are indistinguishable when comparing a pair of sequences. However, in a pro-
gressive alignment, a deletion is only penalised once at the place it occurs, but an insertion has to
be handled at every alignment performed between its original occurrence and the root of the guide
tree. Löytynoja and Goldman (2008) created a new alignment program, PRANK, to use information
from related sequences to indicate whether a gap was due to an insertion or a deletion, to prevent
insertions from being matched in subsequent alignments and to prevent the multiple penalisation of
inserted residues. The authors term this approach phylogeny-aware gap placement. Löytynoja et al.
(2012) extend this work in a new alignment program PAGAN, which uses a partial order graph (Lee
et al., 2002) rather than the more greedy approach for determining an insertion used in PRANK. PA-
GAN does not generate a guide tree itself, but follows the alignment order specified in an externally
defined guide tree.

1.3 Guide trees

By using only pairwise alignments, progressive alignment bypasses the considerable time and mem-
ory required to process the multidimensional dynamic programming matrix described above for an
all-versus-all alignment. Thus, instead of a single alignment step involving all \( N \) sequences, pro-
gressive alignment will have \( N - 1 \) pairwise alignments, thereby removing the exponential aspect of
the complexity measures. These pairwise alignments will be between either a pair of sequences, a
sequence and a profile, or two profiles, and the order in which the alignments occur will follow the
branching order of a guide tree that has been created based on a comparison between each pair of
sequences to be aligned.

While progressive alignment makes the alignment of many thousands of sequences feasible, it
does introduce a number of additional considerations. Firstly, as mentioned earlier, this is a heuristic
approach and the alignment produced may not be optimal but will, hopefully, be sufficiently reliable to
be used in subsequent analyses — how the quality and reliability of the alignment can be determined
is described in the next section. It is a “greedy” algorithm, and alignment errors are not corrected at
later stages. In addition, progressive alignment is specifically a series of pairwise alignments, and so the guide tree that is constructed only contains bifurcations. Even if there are many sequences or profiles that may be aligned at a particular node in the guide tree based on identical comparison results, an arbitrary decision is made and the binary structure of the guide tree is enforced.

The construction of the guide tree typically requires all sequences to be compared with all others, so time obviously scales at least at $O(N^2)$. In addition, the results of the pairwise comparisons have to be analysed to infer the guide tree structure. Typically, more similar sequences are clustered together, starting with individual pairs of sequences represented by the leaves on the guide tree, and continuing into larger and larger groups until all sequences are included in the group at the root of the tree. This clustering is a computationally expensive step, and the more common approaches, UPGMA (Sokal and Michener, 1958) and Neighbour-Joining (Saitou and Nei, 1987) naively scale as $O(N^3)$, but can be implemented as $O(N^2)$ (Press et al., 2007). Even with this improvement, for larger alignments, the guide tree creation may become a limiting factor.

Finally, the order in which the pairwise alignments are carried out, which is determined by the branching order of the guide tree, has a significant affect on the multiple alignment produced. That the alignment order has an impact on the alignment produced has been one of the fundamental tenets of progressive alignment since it was initially developed — if not, why go to the computational expense of creating the guide tree? It is readily demonstrable that the order in which the alignments are carried out does have a marked impact. Penn et al. (2010) put this to good use by showing that perturbing the guide tree can change the resulting alignment, and using a collection of trees can give an indication of the robustness of individual alignment columns. However, the question of what makes a “good” guide tree is not as straightforward, and in part depends on the use to which the alignment is put. More details on this will be presented in Section 1.4 below.

1.3.1 Guide trees versus phylogenetic trees

In the original progressive alignment paper, Hogeweg and Hesper (1984) argued that the generation of a multiple sequence alignment and phyletic (phylogenetic) tree construction are closely related. They continued that phyletic trees produced by matrix methods (methods that use a triangular matrix of the overall pairwise similarity between sequences) are binary trees, and that the alignments of similar sequences are the most reliable. In their view, the only rationale for generating an alignment was to determine the phylogeny of the particular sequences, or as Waterman et al. (1976, p367)
put it: “Biology seeks to discover the evolutionary relations among the set of proteins”. Hence, constructing a guide tree so that most similar sequences are aligned together minimises the number of mutations or indels along the branches of the guide tree, and follows the principle of minimum evolution or maximum parsimony.

Hogeweg and Hesper (1984) proposed an iterative approach to align a set of sequences by using a series of successive pairwise alignments that followed the branching of a hypothesised tree, and to optimise the hypothesised tree and the alignment iteratively by using the previous tree or alignment to construct the next alignment or tree. The end result of this process was both an alignment and a phylogenetic tree. Higgins et al. (1992) only labelled this tree a ‘guide tree’ a number of years later, and over the years the differentiation between the two different types of trees has not always been clear. In a review of the field, Thompson et al. (2011, p1) describes progressive alignment as building up the alignment using pairwise alignments “following the branching order in a phylogenetic tree”. However, Edgar (2004b, p1792) argues that “…assuming that in progressive alignment, the best accuracy is obtained at each node by aligning the two profiles that have fewest differences, even if they are not evolutionary neighbors”. Wheeler and Kececioglu (2007) in their tests confirm that UPGMA or Minimum Spanning Trees (MST) give better alignment results than Neighbour-Joining, even though Neighbour-Joining is considered the best distance-based method at producing the true evolutionary tree from sequences. UPGMA and MST are similar approaches and only differ in how they define the distance from a merged group to all other groups: UPGMA uses the average distance from the merged elements, while MST uses the minimum.

More recent developments have separated the generation of a multiple sequence alignment and the construction of a phylogenetic tree. This is partly due to the numerous different uses of a multiple sequence alignment outlined previously, and in these cases the best guide tree should be the one that produces the best results in the downstream analysis. Also, a number of new approaches have been developed for constructing phylogenetic trees, e.g. maximum likelihood as implemented in RAxML (Stamatakis et al., 2004), all of which work from a multiple sequence alignment. As Price et al. (2009, p1641) state “sequences are identified as homologous and aligned, and then a phylogeny is inferred”. Hence, guide trees should be seen as a necessary intermediate construct in a progressive alignment, and not as a proxy for a phylogenetic tree.
1.3.2 Similarity calculations

Most progressive alignment approaches follow the approach of Hogeweg and Hesper (1984) that more similar sequences should be aligned first. The question then is how to determine which sequences are most similar. As mentioned previously, the time required for the pairwise comparison of all sequences will scale as at least $O(N^2)$. Hence, the method of comparison needs to be efficient as well as accurate.

One of the most common methods uses $k$-tuple scores. The FASTA algorithm (Wilbur and Lipman, 1983) finds the maximum number of matching (possibly overlapping) $k$-tuples of sequence residues when comparing two sequences. The popular progressive alignment programs Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle all use this approach — Clustal Omega uses 1-tuples for proteins, and both Mafft and Muscle use 6-tuples. The $k$-tuple score is usually scaled by the length of the shorter alignment to allow for differing length sequences, and then used as a measure of the similarity of the pair of sequences in question. While this approach is fast, it only scores exact matches of length $k$ between the two sequences, and both the residues that are matched and the position of the matches are ignored. Hence, the similarity score generated should be seen as being a rough approximation of sequence similarity. The shortcomings of these similarity measures will be examined in more detail in Chapter 3.

There are a number of variations on this approach including approximate string matching (Wu and Manber, 1992) as implemented in Kalign 1 (Lassmann and Sonnhammer, 2005a), string matching with one error (Muth and Manber, 1996) used in Kalign 2 (Lassmann et al., 2009), exact matches of variable length (Ulitsky et al., 2006), using spaced patterns of “match” and “don’t match” positions (Ma et al., 2002), and using multiple spaced-word patterns (Leimeister et al., 2014). Another measure is the edit distance between two sequences, originally proposed by Levenshtein (1966). This is defined as the minimum number of edit operations — insertions, deletions and substitutions — needed to transform one string into another. (Note this is a measure of distance between sequences, rather than similarity, with higher values indicating that sequences are less similar to each other.) However, while most of these methods tend to be reasonably fast, they all suffer from the same issues as $k$-tuple scores.

A number of points are worth noting. Firstly, except for the edit distance, the comparison operations described above give a measure of the similarity between each pair of sequences, with more similar sequences having higher scores. These are usually converted into a distance mea-
sure, where more similar sequences are “closer” to each other and so have a smaller distance value. With Clustal Omega, for instance, the scaled \( k \)-tuple score is subtracted from 1.0 to convert it into a distance measure in the range between 0.0 (exactly matched sequences) and 1.0 (completely dissimilar sequences). The collection of all pairwise distances is usually referred to as the distance matrix, where the value in \( D_{ij} \) is the distance measure between the \( i \)-th and \( j \)-th sequences. This distance matrix must satisfy three properties. It must be symmetric (for all \( i \) and \( j \), \( D_{i,j} = D_{j,i} \)), non-negative (for all \( i \) and \( j \), \( D_{i,j} \geq 0 \)) and satisfy the triangle inequality (for all \( i, j \) and \( k \), \( D_{i,j} + D_{j,k} \geq D_{i,k} \)). Finally, from a performance point of view, the similarity calculation step in a progressive multiple sequence alignment is the most readily optimised through parallelisation, as each of the comparison operations between the pairs of sequences are typically carried out independently.

### 1.3.3 Clustering

Once all pairwise distance measures have been calculated, a hierarchical clustering algorithm uses these measures to group the sequences and make the guide tree. Starting with the individual sequences represented by the leaves of this tree, each higher-level node of the tree combines two lower-level nodes or leaves until the root of the tree is reached which includes all sequences to be aligned. By only combining two nodes or leaves at each level, a strictly bifurcated tree is produced. Hence, by following the branching order of this tree, the alignment steps will align the most similar sequences together and each alignment will be a pairwise alignment.

The two most commonly used clustering algorithms in progressive alignments are UPGMA and Neighbour-Joining. Both are agglomerative methods, starting with \( N \) trivial clusters represented by the leaves in the guide tree, each containing one sequence to be aligned, and then combining these clusters into larger clusters, and so forth. UPGMA (Unweighted Pair Group Method with Arithmetic Mean) finds a pair of “closest” clusters, where the distance between clusters is defined as the average pairwise distance between the elements of the clusters. Once the closest clusters have been found, a new cluster is formed at a higher level in the guide tree containing the elements of the closest clusters, and the distance of this new cluster to all other clusters is calculated as the average distance of all the merged elements. Neighbour-Joining finds a pair of neighbouring leaves in the tree and replaces them by a single leaf, and can thus recursively construct a tree from the distance matrix. This tree is by default an unrooted tree, but this can easily be converted into a rooted tree.
by replacing one edge between nodes $u$ and $v$ with a root and connect the root to each of $u$ and $v$. A common approach is to use mid-point rooting, where the root is placed midway along the longest tip-to-tip path.

UPGMA was originally proposed as a way of forming generic classification hierarchies, whereas Neighbour-Joining was specifically intended for constructing phylogenetic trees. Both methods can construct ultrametric trees, assuming an accurate distance matrix. However, UPGMA makes the invalid assumption that the smallest element in the distance matrix corresponds to a pair of neighbouring leaves in a phylogenetic tree. Neighbour-Joining, on the other hand, transforms the distance matrix at each step so that the minimum element does yield a pair of neighbours. In addition, Neighbour-Joining performs well in real-world situations where the distance matrix is not fully additive, in that the sum of the path lengths between any two leaves in the tree through their closest common ancestor does not correspond to the values in the distance matrix. A simple test for an additive matrix is given by the four-point condition, that the two largest of $(D_{ij} + D_{kl}), (D_{ik} + D_{ji})$ and $(D_{il} + D_{jk})$ are equal for all $i, j, k$ and $l$, where $D_{ij}$ is again defined as the distance measure between the $i$-th and $j$-th sequences (De Soete, 1983). It is for these reason that Neighbour-Joining is widely used for evolutionary tree reconstruction. However, in benchmarking tests, alignments using guide trees created with UPGMA tends to be better than those created using Neighbour-Joining. Sievers et al. (2014) compare numerous clustering schemes for small datasets for Clustal Omega, Muscle, PAGAN, and both FFT-NS-1 and L-INS-1 for Mafft, and in each case UPGMA outperformed Neighbour-Joining. Edgar (2004b, p1792) also uses UPGMA for clustering “which we find to give slightly improved results over neighbor-joining”. Wheeler and Kececioglu (2007) used three standard protein alignment benchmarks to compare the accuracy of the alignments produced using different clustering approaches. Again for all three benchmarks alignments using UPGMA guide trees had higher accuracy scores than alignments created using Neighbour-Joining guide trees. Wheeler and Kececioglu believe that this may be due to the point that Edgar (2004b) makes, that using a merge tree similar to the correct evolutionary tree is less important than using a tree that groups similar sequences first.

As shown previously, the calculation of sequence similarity and the subsequent clustering both have a time complexity of at least $O(N^2)$, making the construction of the guide tree potentially excessive when aligning large numbers of sequences. Among the approaches that have been proposed to reduce this time complexity is the PartTree algorithm (Katoh and Toh, 2007), which has been imple-
mented in Mafft. This constructs an approximate guide tree using a recursive clustering algorithm, which include a random selection of “seed” sequences at each recursion level around which the other sequences are clustered. The recursion continues until the clusters fall below a certain size, at which point the trees at each level are combined into one overall tree. The algorithm is reported to have an average time complexity of $O(N \log N)$ and a space complexity of $O(N)$. However, this speed improvement comes at a cost of several percent in alignment accuracy, as benchmarked on the Pfam database of aligned protein families (Finn et al., 2006).

Another approach to improve the time requirements of the guide tree creation is the mBed algorithm (Blackshields et al., 2010), used by default in Clustal Omega. A selection of $\log^2 N$ “seed” sequences are made using a constant stride from the length-sorted $N$ input sequences. The distances from these seed sequences to all input sequences is then calculated. These distances are then clustered by repeatedly applying $k$-means clustering with $k = 2$ to bisect the set of sequences until each cluster contains 100 (default value) or fewer sequences. Once these clusters have been identified, all distances between each pair of sequences in a cluster are calculated. A UPGMA tree is created for each cluster, and these sub-trees are then combined into an overall tree. While there are a number of different steps in the mBed algorithm, these are carried out sequentially and work with far smaller datasets, and so give a considerable speedup over the standard approach of calculating the similarities between each pair of sequences to be aligned.

In addition to clustering the sequences to create the guide tree topology, the UPGMA and Neighbour-Joining algorithms also assign a non-negative length to each of the branches of the guide tree. In phylogenetic trees, these branch lengths correspond to evolutionary time whereas in guide trees the branch length is more an indicator of the similarity of the subtree to the internal node. In progressive alignment, given that the distance measures are typically calculated from $k$-tuple scores and that most similar sequences are aligned first, the guide-tree branch lengths would seem to be of minor importance. Clustal Omega ignores these branch lengths, and while Mafft does take them into consideration, there is a command-line option (--unweight) to ignore them. Sievers et al. (2014) found that the branch lengths had no effect on the quality of the alignments produced.

1.3.4 Iterations

Progressive alignment is a greedy alignment, and errors made in early alignments can accumulate and degrade the overall alignment. The two basic mechanisms for dealing with such errors are to
try and prevent them occurring in the first place by using consistency-based alignment as described previously, or to try and fix the errors afterwards. This latter approach is what Wheeler and Keciciglu (2007, i564) term “polishing the alignment”, or what is more commonly referred to as an iteration strategy (Hirosawa et al., 1995). In general, the initial alignment is produced using a guide tree constructed on the $k$-tuple score described above. The alignment is then refined either by regenerating the guide tree based on the similarity between pairs of sequences in the initial alignment, or by manipulating the guide tree.

Barton and Sternberg (1987, p328) were among the first to propose alignment iteration, in which the input sequences were first added one at a time to the growing alignment, and the alignment produced was then “refined by realigning each sequence with the completed alignment less that sequence”. Berger and Munson (1991) used randomised iterations, in which the unaligned sequences are randomly divided into two groups, which are then rejoined by dynamic programming. This pairwise alignment process is then repeated using different random divisions of all the sequences into two groups. Gotoh (1996) compared Clustal W with three randomised iterative methods, and found the iterative methods to be more accurate than the conventional progressive alignment method of Clustal W, and this improvement in accuracy tended to increase for more distantly-related sequences with lower average sequence identity. Interestingly, the alignment accuracy was assessed by referring to a subset of sequences with known structures included in each alignment, and the alignment of these sequences was compared to their structural alignment in a residue-to-residue fashion, in the same way as Barton and Sternberg (1987) defined the accuracy of their alignments. Do et al. (2005) also included iterations as a post-alignment step in ProbCons, and it has been included as an option in most multiple alignment programs since then.

Clustal Omega has two iteration modes, guide-tree iteration or HMM iteration, both of which can be invoked independently. To recreate the guide tree, Clustal Omega by default uses the number of matching columns in each pair of sequences as a measure of the similarity of those sequences, the idea being that this should be more meaningful than the original $k$-tuple score and a guide tree produced from these aligned sequences should result in a better alignment. Iterating the HMM or the alignment step attempts to get around the greedy nature of progressive alignment. The initial alignment is converted to a profile HMM, and individual sequences and small profiles are then aligned with the profile HMM. The results (Sievers et al., 2013) indicate that a small number of iterations does yield a better alignment quality when measured against structural reference sequences, with
HMM iteration appearing to perform better than guide tree iteration.

Muscle has a more detailed strategy. As mentioned above, the initial guide tree is constructed based on shared 6-tuples between each pair of unaligned sequences. If a second iteration is requested, while the fractional identity between two sequences is often used as a similarity measure, a Kimura correction (Kimura, 1984) is made to this to allow for more divergent sequences having multiple mutations at a single site (Edgar, 2004a). The Kimura distance matrix is then clustered as before and a new alignment generated. By default Muscle will attempt up to 14 more refinement iterations, but will terminate early if no changes have been made after one full iteration. In each refinement iteration the guide tree is split at every branch, starting with those furthest from the root, which divides the alignment into two profiles, a technique called tree-dependent restricted partitioning (Hirosawa et al., 1995). The two profiles are re-aligned using Muscle’s profile-profile alignment, and this alignment is kept if the alignment’s SP score is improved.

Mafft adopts a similar approach to Muscle. The initial guide tree used in the FFT-NS-1 algorithm is also based on shared 6-tuples between each pair of sequences. In more recent versions of Mafft, this score is scaled based on the lengths of the two sequences, as otherwise a very short and a very long sequence may have a distance close to zero by chance even when the sequences are unrelated. In the FFT-NS-2 algorithm, the input sequences are realigned using a guide tree inferred from the first alignment. For subsequent iterations, one edge chosen at random is split rather than cycling over all edges as in Muscle. Again the resulting two profiles are re-aligned, and the process continues for the specified number of iterations (the default is 1,000) or until no better scoring alignment is obtained.

1.4 Benchmarking

Multiple sequence alignment has become the typical first step in a wide variety of comparative analysis workflows, many of which have competing objectives. Over the years, numerous alignment programs have been developed — Kemena and Notredame (2009) estimate more than 100 — each with its own targeted sequence type, optimisation technique, or other speciality. A seemingly obvious, but apparently often overlooked, question then is whether the multiple sequence alignment program being used in a particular pipeline is the most appropriate one. Chatzou et al. (2015) point out that Clustal W’s citation count suggests a usage level higher than all other alignment programs combined, even though it has not been consistently reported as generating the most accurate align-
ments (Sievers et al., 2011; Thompson et al., 2011). Chatzou et al. put this down to two factors: Clustal W is sufficiently accurate for many modelling activities, particularly when dealing with orthologous data sets, and also possibly due to “historical inertia” (Morrison, 2009, p151) in which existing protocols tend to be reused when new workflows and analysis projects are being developed.

Iantorno et al. (2014) argue that multiple sequence alignment methods need to be objectively validated to ensure the alignments they produce are both accurate and reproducible, and that benchmarking against a known solution provides a way of evaluating the different alignment programs objectively. Depending on the purpose for generating the multiple alignment, a benchmark specifically aimed at that usage can be chosen thereby giving the users a degree of confidence that their chosen alignment program is adequate for their needs. Kemena and Notredame (2009) show that the ranking of various alignment programs differs considerably depending on whether simulated or structure-based benchmarks were used. Interestingly, Blackburne and Whelan (2013) examined a number of multiple sequence alignment methods and found that they divided into two distinct classes: a similarity-based class which included progressive and consistency aligners, and an evolution-based class containing the phylogenetically aware alignment programs. They also found that the class of alignment program used had an impact on different downstream analyses. Batzoglou (2005) points out that there are two biologically meaningful goals of alignments: the desire to find evolutionary relationships and the desire to find putative functional relationships. It seems that both goals may require different benchmark tests.

Iantorno et al. (2014) outline four different benchmarking strategies, which are described next.

1.4.1 Consistency among alignment methods

Unfortunately, this use of the term consistency differs from that of consistency-based alignment programs (T-Coffee, etc.) described previously. Here, consistency refers to the similarity between alternative alignments of a set of sequences, sometimes created by different alignment programs. The consistency-based alignment programs, on the other hand, are aiming to achieve consistency between pairwise alignments within a single multiple alignment. Lassmann and Sonnhammer (2005b) refer to these as inter-consistency and intra-consistency respectively.

Mevissen and Vingron (1996) used the concept of a “robustness measure” to indicate which residues in an alignment are more likely to be reliably aligned. Chatzou et al. (2015, p11) point out that “any procedure that may be used to perturbate an alignment lends itself to the definition of a
One of the simplest such approaches is the Heads-or-Tails check (Landan and Graur, 2007). Given a set of sequences to be aligned (the heads set), an alternative set (the tails set) is created by reversing the residue order in each sequence. Both sets are then aligned, and the degree of agreement between the two sets of results can be used to assess the reliability of the alignment. Penn et al.’s (2010) approach of perturbing the guide tree was introduced previously, including using a collection of trees to indicate the robustness of individual aligned residues. They found that this method, named GUIDANCE, was a better predictor of unreliably aligned regions than the Heads-or-Tails method. Another more computationally-expensive method, PSAR, was proposed by Kim and Ma (2011). In this, one sequence at a time is extracted from the supplied alignment, and the sequence and sub-alignment compared using a pair-HMM. A probabilistic model is then created and used to generate alternative alignments, and the level of agreement between these alternative alignments and the original alignment can be used to calculate a reliability score for each residue in the original alignment.

1.4.2 Simulated sequences

Benchmarks based on simulated sequences are widely used for assessing alignment programs that are used for phylogenetic modelling. As the full evolutionary history is known, alignments generated from the simulated sequences by different alignment programs can be compared with the ‘true’ reference alignment, something that is not always known with certainty when using real sequences. From this comparison, different accuracy measures can be calculated, the two most common being the Sum-of-Pairs (SP) score and the Total Column (TC) score (Thompson et al., 2005). The SP score is the fraction of correctly aligned pairs of residues in the alignment, and determines how well the alignment program aligns some if not all of the sequences. The TC score is the fraction of correctly aligned columns in the generated alignment as compared with the correct alignment. As a single misaligned sequence can nullify the TC score, the SP score is frequently used for more divergent sequences. However, as Blackburne and Whelan (2012) point out, neither the TC nor the SP scores are true metrics, as they violate both the principles of symmetry and triangular inequality. Nevertheless, both the SP and TC scores remain the most common measurements for determining the quality of an alignment.

There are numerous packages that can generate simulated sequences, including Rose (Stoye et al., 1998), DAWG (Cartwright, 2005) and ALF (Dalquen et al., 2012). These use probabilistic
models of sequence evolution to cater for substitution, insertions and deletions, gene duplications, and other speciation events. The simulations usually continue until a specified number of different sequences have been generated, or a particular length of evolutionary time has elapsed.

In addition to knowing the ‘true’ alignment, simulations of this nature have two other main advantages. Firstly, the generation of additional data for testing larger alignments is trivial, and the knowledge that all sequences have been generated using the same underlying models is reassuring. The other advantage can be seen as a double-edged sword. The flexibility of the simulation environment allows the generation of sequences targeting very specific situations or alignment programs. However, this flexibility also means that sequences can be generated that may bear little resemblance to real biological data, as the assumptions and simplifications underlying the simulation environment may not take all evolutionary constraints into consideration. As an example, most simulation programs only deal with a protein’s primary structure. Constraints based on its secondary or tertiary structures, the tendency of indels to be restricted to loops and coils rather than helices and strands (Pascarella and Argos, 1992), or the co-evolution of residues adjacent in 3D space (Gromiha and Selvaraj, 2004) are usually ignored. This point will be discussed again in Section 2.6.

1.4.3 Phylogenetic tests

These benchmarking tests construct a phylogenetic tree (by maximum likelihood) from the given multiple sequence alignment, the assumption being that “the more accurate the resulting trees, the more accurate the alignments (in terms of homology matching) are assumed to be” (Dessimoz and Gil, 2010, p2). Dessimoz and Gil propose two specific tests. The first, “species-tree discordance”, uses only orthologous genes and compares the generated phylogenetic tree with the known species tree, and the more accurate alignment programs will more often result in trees congruent with the known tree. The second test, “minimum duplication”, works with homologous sequences and is based on maximum parsimony rather than knowing the true phylogeny. The sequence alignment method that results in trees with fewer duplications is considered to be the better method. Taken together, both of these tests can be performed on virtually any set of sequences, and so can be used to select the best alignment program for the study in question.

As Iantorno et al. (2014) point out, a key advantage of these tests is that they can evaluate gap-rich and variable regions in sequences, which are typically not catered for in structural benchmarks or for which simulation benchmarks lack realism.
1.4.4 Structural benchmarks

These benchmarks have been developed from known protein structures, and are the most widely-used means of determining the quality of different alignment programs (Blackshields et al., 2006). Typically, they use the superposition of the known structures as an independent reference alignment against which the proposed alignment can be compared using either the SP or TC scores described above. While it is to be expected that these benchmarks would be relevant for structure and function predictions, they are also used to assess alignments for other functions as the reference sets are based on experimentally-derived structural information and so are seen as being independent empirical benchmarks. Also, while proteins with similar sequences adopt similar structures, structures are usually more conserved than the underlying sequences (Chothia and Lesk, 1986; Rost, 1999). Proteins with only weakly similar sequences can have a highly similar 3D structure (Holm and Sander, 1994). Hence, alignments based on structural similarity may be more reliable for highly divergent sequences.

HOMSTRAD (Mizuguchi et al., 1998b) is a database of aligned three-dimensional structures of homologous proteins. The structures are aligned using a variety of automated tools, including MNYFIT (Sutcliffe et al., 1987) and COMPARER (Săli and Blundell, 1990), and the resulting alignments are manually checked. The various structural features are also annotated using JOY (Mizuguchi et al., 1998a). It was not intended to be used as a multiple sequence alignment benchmarking dataset, but it has been used to assess the quality of different alignment programs (Edgar, 2010). Its contents have also been incorporated into some of the other benchmark datasets described below.

BAliBASE (Thompson et al., 2005) was one of the first specifically designed structural benchmark dataset. The current version, 3.0, comprises 6,255 sequences in 217 reference alignments across 5 Reference sets, with each Reference set aimed at a particular alignment challenge: sequences with a particular range of percentage identity, sequences with large N/C-terminal extensions, sequences from different sub-families, etc. The benchmark construction is a two-phase process. In the first phase, sequences in the previous version of BAliBASE are used to search the PDB database (Berman et al., 2000) using PSI-BLAST (Altschul et al., 1997). In addition, new families are added by manually selecting sequences from SCOP (Andreeva et al., 2004) and these are used to search the PDB database. Sequences with >40% identity are removed and the remaining non-redundant sequences are aligned using the SAP 3D superposition program (Taylor, 2000). The output is then manually curated to ensure the alignment of conserved residues. The second phase
expands the test cases by including sequences of unknown structure from the Uniprot database (Apweiler et al., 2004). Sequences are included if they share >40% identity with at least one PDB sequence, as divergent sequences may not be accurately aligned without structural information as a guide. In addition, sequences with >80% identity are considered redundant and removed. The remaining sequences are then aligned and again manually verified to correct for bad alignments.

To compare the relative performance of a number of alignment programs, each program aligns the unaligned BAliBASE sequences, and the SP and TC scores can be calculated by comparing the alignments produced with the reference alignments included with BAliBASE. The SP and TC scores can be calculated on all columns in an alignment or only on “core blocks”, which are areas in the alignment that are considered to be reliably aligned.

While BAliBASE is widely used, Edgar (2010) has highlighted a number of issues. In his analysis Edgar finds that, when compared to the SCOP and CATH (Orengo et al., 1997) structure databases 87% of BAliBASE sequences do not have a known structure. In addition, these sequences were aligned based on their primary sequence alone, which may introduce bias into quality measures reported from BAliBASE if the alignment program being benchmarked uses similar parameters to those used to construct the reference alignments. In a separate analysis, Edgar (2009) indicates that the BLOSUM62 substitution matrix might be favoured. Finally, Edgar points out that many non-core-block columns in the BAliBASE reference alignments contain non-homologous domains, and therefore should be considered as an incorrect alignment but in practice are often included in the reported quality scores. However, as Aniba et al. (2010) point out, this ignores the fact that a single amino acid change can lead to a large change in the protein structure, as shown by Meier et al. (2007).

To prevent the inclusion of unknown structures in the reference alignments, the PREFAB (Edgar, 2004b) test cases consist of two proteins sequences that are aligned by a structural method without using sequence similarity. Both sequences are then used to query a protein database in order to retrieve high-scoring homologues. The retrieved sequences are filtered to ensure a maximum of 80% identity, 24 of the remaining sequences are selected at random for each query sequence, and combined with the original sequences to give a test case with at most 50 sequences. The test cases are aligned by the program being tested, and the alignment accuracy determined with reference to the structural alignment of the original pair of sequences.

OXBench (Raghava et al., 2003) uses a structural alignment created by the STAMP structural
alignment program (Russell and Barton, 1992) on proteins drawn from the 3Dee structural domain database (Siddiqui et al., 2001), which contains domain definitions of proteins with experimentally-determined 3D structures in the PDB. In addition to the actual alignment, STAMP also calculates a structural similarity score $S_C$, which gives a measure of the reliability of each structurally-aligned position and which can be used to filter out ambiguous alignments. Like PREFAB, the construction of OXBench is fully automated and the reference alignments are generated only using a structural alignment method.

SABmark (Van Walle et al., 2005) consists only of sequences with very low to intermediate similarities (0–50% identity), as these tend to be the most difficult to align. In addition, this also allows for benchmarking the alignment of unrelated sequences. All sequences have a known structure, and are systematically selected to ensure all known folds are included. The sequences are divided into two subsets, Twilight Zone and Superfamilies, which includes sequences with very low to low, and low to intermediate, similarity respectively. This Twilight Zone differs from that of Rost (1999). Pairwise reference structure alignments are constructed using a consensus of the alignments generated by CE (Shindyalov and Bourne, 1998) and SOFI (Boutonnet et al., 1995). These pairwise alignments will not be consistent across all sequences, which reflects the uncertainty about parts of the reference alignment or the multiple possible solutions to measuring structural similarity (Lackner et al., 2000).

All of the test cases in the benchmarks so far are of reasonably modest size, e.g. the largest alignment in BAliBASE has 142 sequences. The focus of this thesis is on the behaviour of the existing alignment algorithms when generating large-scale alignments. HomFam (Blackshields et al., 2010) combines HOMSTRAD single-domain protein families with at least 5 sequences with sequences from the corresponding homologous Pfam domain. Residues are marked as core in the reference alignments if the JOY-annotated structural features agree in at least 70% of the reference alignments, and the alignment quality can be assessed using either the SP or TC scores on either all columns or just the core columns. This approach provides large numbers of sequences to be aligned — the rvp family (Retroviral aspartyl protease, Pfam accession number PF00077) currently has over 200,000 sequences. However, the alignment quality is only measured using the HOMSTRAD sequences embedded in the alignment, which is typically in the range of 5–15 reference sequences per family. While HomFam allows a reasonable assessment of the time performance of different alignment programs on large datasets, its use for quality assessment should be treated
with caution. The two main assumptions underlying the approach are that the reference sequences are representative of the rest of the sequences, which may become questionable as the number of sequences becomes very large. In addition, the dispersion of the reference sequences throughout the guide tree can influence the quality measures. HomFam is only relevant for large alignments, which will invariably be made using a progressive alignment approach. Once the reference sequences have all been aligned, aligning subsequent sequences has no effect unless the complete alignment is iterated, as the position of the reference sequences’ residues relative to each other will not change. Hence, if all the reference sequences are located on one side of a reasonably-balanced guide tree, approximately half the sequences will have no effect on the quality of the alignment produced. In addition, replacing those sequences with more distantly-related (or completely unrelated) sequences will not change the quality of the overall alignment if the reference sequences remain in the same branches of the guide tree.

Like HomFam, ContTest (Fox et al., 2016) allows the assessment of large-scale multiple sequence alignments, with no theoretical upper limit to the size of their alignments. It relies on recent methods to predict protein structure information directly from an alignment of the sequence of interest with a large number of homologous sequences (Kamisetty et al., 2013), as demonstrated in the programs EVfold (Marks et al., 2012) and PSICOV (Jones et al., 2012). Residues in contact in a folded protein may co-evolve so that a mutation in one position corresponds to complementary mutations in nearby residues. Both EVfold and PSICOV generate a list of pairs of residues they predict to be in contact based on the correlations they have detected between substitutions in pairs of columns in an alignment, and these pairs of residues can be used to generate a contact map. ContTest includes one protein sequence that has a known 3D structure in the alignment, and compares the predicted contact map generated from the alignment with the protein’s known contact map. The accuracy of the alignment is then calculated based on the precision of the top $L/5$ long-range contacts predicted, where $L$ is the length of the known protein’s sequence and long-range means contacts are separated by at least 23 residues. Unlike under some scenarios with HomFam, ContTest uses all sequences in the alignment. It also avoids the superposition of predicted and known structures which uses a heuristic approach. However, it also assumes that the embedded protein is representative of the sequences being aligned, and it does rely on the accuracy of the EVfold and PSICOV predictions.

An underlying issue with many of the structural benchmarks is the inclusion of potential bias in
their means of construction. While manual curation is believed to give more biologically accurate benchmark sets than purely automated procedures, this might also introduce a subjective bias into the alignments (Iantorno et al., 2014). However, purely automated methods still depend on arbitrary parameter values, and these parameter values may have been selected for computational efficiency reasons or may not be appropriate for all datasets. In addition, there is an inherent bias in the types of proteins included in the structural benchmarks: “Current MSA programs for protein sequences generally model globular domain structure and evolution. Nonetheless, many proteins, particularly in eukaryotes, are unstructured (natively disordered) or contain large unstructured regions.” (Thompson et al., 2011, p2).

There is also a danger that a particular alignment program is over-fitted for one specific test, but this can be prevented by measuring performance against two separate benchmarks. Blackshields et al. (2006) and Kemena and Notredame (2009) found that empirical reference datasets tend to give similar results across alignment programs.

1.5 Large alignments

Finally, some additional comments on large alignments, which in the context of this thesis is in excess of about one thousand sequences. Some alignment programs use different algorithms or parameter settings when aligning large numbers of sequences, e.g. clustering using PartTree in Mafft, or reducing the number of iterations in Muscle. These modifications to the typical approaches are deemed necessary as the standard applications would otherwise require too much time to complete.

There are also, however, algorithms and alignment programs specifically designed to work with datasets consisting of tens of thousands of sequences. Among these are FastTree (Price et al., 2009), used to compute large minimum evolution phylogeny trees. For multiple sequence alignments, the program FAMSA (Deorowicz, Debudaj-Grabysz and Gudys, in press) aligns over 400,000 sequences on a standard desktop computer.

Another such alignment program is UPP (Nguyen et al., 2015), which uses an ensemble of HMMs to create the overall alignment. UPP preferentially selects a subset of the unaligned sequences, based on the length of the sequences. These sequences are used to create an alignment (the “backbone alignment”) and unrooted maximum-likelihood tree (“backbone tree”) using PASTA (Mirarab et al., 2015). UPP creates a collection of HMM profiles by first creating an HMM profile from the backbone alignment using HMMER (HMMER). The backbone tree is decomposed by re-
moving a centroid edge (i.e. an edge that divides the tree into two approximately equal subtrees) and HMM profiles are created for both subtrees and these are added to the HMM profile collection. The decomposition of subtrees and creating of profiles continues until each subtree contains at most 10 sequences. Those sequences not selected to create the backbone alignment are then inserted into the subset alignment with which they have the best fit, as measured using the HMMER bitscore returned when a sequence is queried against the collection of HMM profiles. The subset alignments are then combined to produce the overall alignment. Using parameter settings for maximum speed, UPP was able to align 200,000 RNASim (Mirarab et al., 2015) sequences on a 12-core computer with 256 GB RAM in under 24 hours.

One final aspect of large-scale alignments should be noted. Sievers et al. (2013, p989) benchmarked a wide-range of multiple sequence alignment programs with datasets of various sizes and found “the accuracy of such alignments decreases markedly as the number of sequences grows. This is more or less true of all packages and protein families.” Figure 1.3 summarises their results. This fall-off in alignment quality is one of the main topics that this thesis attempts to address.
Chapter 2

Chained Guide Trees


With the exception of Figure 2.14, reproduced from Tan et al. (2015), Figure 2.16 and the utility programs to construct the guide trees and calculate the guide tree metrics described in Table 2.1, all created by Fabian Sievers, I conducted the research presented here and analysed all data produced.

2.1 Introduction

As outlined in Chapter 1, the first step in progressive multiple sequence alignment involves clustering the sequences to be aligned into a tree or dendrogram-like structure, called a “guide tree” in Higgins et al. (1992). The sequences are then aligned into progressively larger and larger alignments following the branching order in the guide tree. Variations on this method were described by different groups in the 1980s (e.g. Taylor (1988); Barton and Sternberg (1987)) but the earliest clear description of the approach is from Hogeweg and Hesper (1984). Progressive alignment is a heuristic approach and is not guaranteed to find the best possible alignment for any given scoring scheme. It does, however, allow alignments of many sequences to be made quickly, even on what are currently considered to be modest personal computers (Higgins and Sharp, 1988). The quality of the alignments produced is then good enough for the alignments to be used automatically in many analysis pipelines.
The computational complexity of a multiple sequence alignment comprises three parts: the calculation of the similarity score between each pair of sequences, requiring \(O(N^2)\) time and computer memory, where \(N\) is the number of sequences being aligned; the clustering of these similarity scores into a guide tree, usually requiring \(O(N^2)\) or \(O(N^3)\) for UPGMA (Sokal and Michener, 1958) and Neighbour-Joining (Saitou and Nei, 1987) respectively; and the alignment process itself, which requires approximately \(O(N)\) for similar length sequences. For large \(N\), the construction of the distance matrix and guide tree becomes the limiting factor and prevents the routine alignment of more than a few thousand sequences. More details on the methods used to construct a guide tree, and various approaches to reduce the computational complexity, are given in Section 1.3.

Most of the guide tree construction methods rely on the importance of creating a “good” guide tree with a topology that closely resembles a phylogenetic tree of the sequences, although some suggest that it is better to group similar sequences together than use a correct evolutionary tree (Edgar, 2004b; Do et al., 2005). There seems to have been little work that systematically tested major variations in guide tree topology to measure the effects on alignment quality. Barton and Sternberg (1987) used a simple “chained” guide tree topology, effectively aligning the sequences one at a time to a growing alignment. Taylor (1990) also used chained guide trees to make very large alignments of over 6,000 sequences. Nelesen et al. (2008) looked at some variations in the algorithm used to generate the tree and concluded that there was little influence on the final alignment quality. Wheeler and Kececioglu (2007) compared a number of tree-construction algorithms, including Neighbour-Joining, UPGMA and a new method that recalculates distances as the alignments progress. They found that a minimum spanning tree gave good results. In the phylogenetic tree reconstruction literature, there seems to be a consensus that the guide tree topology should resemble the true phylogeny of the sequences as much as possible (Warnow, 2013).

This chapter looks in detail at the effect of guide tree topology on the quality of protein multiple sequence alignments where the quality of the alignments is empirically measured using protein structure-based benchmarks. The difference in quality is measured when using guide trees ranging from fully “balanced” to completely chained, including intermediate levels of chaining. The ordering of the sequences across these topologies is also examined using, among other approaches, a global minimisation of the distances between sequences and a random ordering. The results are quite surprising in that, for large numbers of sequences (e.g. of the order of one thousand or more), the guide trees that gave the best alignments had completely chained topologies. These had signif-
icantly better alignment scores than “balanced” trees where the topology was either (i) random, (ii) optimised or (iii) the default topology produced by the alignment program. In addition, the accuracy of alignments using chained guide trees was the same, regardless of whether the ordering of the sequences over the guide tree were optimised or were completely random.

2.2 Methods

This chapter examines how different guide tree topologies affect the quality of alignments produced by Clustal Omega (Sievers et al., 2011), Mafft (Katoh et al., 2002) and Muscle (Edgar, 2004b). These programs were selected based on their widespread use, their ability to use an externally-defined guide tree and to align more than a thousand protein sequences. Kalign2 (Lassmann et al., 2009) was also used in one test, but only to examine the guide tree it produced.

The default versions of all aligners were used, with runtime parameters limited to those required to specify the input guide trees. For Mafft, the FFT-NS-2 algorithm was used for all datasets. This is the default algorithm when a particular alignment algorithm is not specified. In addition, the auto flag was not selected, as this causes Mafft to switch automatically between a standard progressive or consistency-based alignment based on the number and length of the input sequences. The newick2mafft.rb ruby script, available from the Mafft website, was used to convert all externally-generated guide trees into Mafft format. With Muscle, the number of iterations was limited to 2 rather than the default of 16. This smaller value is the number of iterations recommended by Muscle’s author when aligning large numbers of sequences. Attempts at running Muscle with the default number of 16 iterations resulted in prohibitive run times and had to be abandoned. Branch lengths are ignored in Clustal Omega and Muscle, and the --unweight option was used in Mafft. The program versions and runtime arguments used are as follows:

**Clustal Omega** (v1.2.0-r289):

- `-i <sequence file> -o <alignment file> --guidetree-in=<guidetree file>`

**Mafft FFT-NS-2** (v7.029b):

- `--anyymbol --treein <guidetree file> --unweight <sequence file> > <alignment file>`

**Muscle** (v3.8.31):

- `--in <sequence file> -out <alignment file> -usetree_nowarn <guidetree file>`
Kalign2 (v2.04):
    -i <sequence file> -o <alignment file> -printtree <guidetree file> -q

In addition to each alignment program's default guide trees, completely chained or "comb-like" guide trees, balanced guide trees, and guide trees with intermediate levels of chaining were created using a separate utility program. These latter guide trees are constructed by removing a given number of sequences from one "side" of a balanced guide tree and replacing that branch with a chained tree structure, resulting in a guide tree that is partly balanced, partly chained. Examples of these types of guide trees, as well as a default guide tree, are given in Figure 2.1.

To test the quality of the alignments produced by each alignment program and guide tree topology, the HomFam alignment benchmark system (Blackshields et al., 2010) and BAliBASE version 3.0 (Thompson et al., 2005) were used. For HomFam, alignments varied in size from 16 sequences up to 32,768 — the number of sequences in each dataset was always a power of 2 as perfectly balanced guide trees can only be constructed from such numbers.

As well as defining the topology of the externally-generated guide trees, it was also necessary to specify the order in which the sequences were allocated to the tips of the trees. The following different sequence orders were used. In each case, all the HOMSTRAD (Mizuguchi et al., 1998b) reference sequences for that family were combined with sufficient sequences selected at random from the HomFam family to give the desired number of sequences to be aligned.

- Random. The input sequences were randomly allocated to the guide-tree leaves.

- TSP Minimum. This was an attempt to minimise the overall distance, or alternatively to maximise the overall similarity, between the sequences to be aligned. The standard Travelling Salesman Problem algorithm using simulated annealing, excluding the distance from the last point back to the starting position, was used to produce an ordered list of sequences with a minimum global distance between the sequences. These sequences were then assigned to the leaves of the guide tree from the left-most leaf of the tree to the right-most leaf. The distances were obtained from the full distance matrix produced by Clustal Omega.

- Greedy Neighbour Minimum. One sequence in the input dataset was selected at random. The closest or most similar sequence to the one just picked was selected, using the distances
Figure 2.1: (a) default guide tree produced by Clustal Omega for a sample of 16 sequences. (b) perfectly balanced and (c) chained guide trees created by a utility program for these same sequences. (d) a guide tree with an intermediate level of chaining created by chaining 4 sequences ‘to the side’ of the balanced guide tree.
from Clustal Omega’s full distance matrix. The process was repeated until all sequences were selected, giving an ordered list of sequences based on local distance minimisation. Again, the sequences were assigned to the leaves of the guide tree starting with the left-most leaf and moving across the tree to the right-most leaf.

- TSP Maximum. This is similar to the TSP Minimum approach, but the ordered list of sequences produced maximises the global distance between the sequences. It was included as a negative test of the effect of sequence ordering on the alignment quality.

Details of utility programs and scripts used are given in Appendix A. All statistical analysis comparing TC scores used the non-parametric one-tailed paired Wilcoxon signed-rank test.

## 2.3 Results

### 2.3.1 Initial analysis

The Cytochrome P450 protein family was used in an initial exploratory analysis, as it has a large number of homologous sequences available in Pfam (Pfam accession number PF00067) and 12 sequences with known 3D structures in HOMSTRAD. As described in Chapter 1, the structure-based alignment of the 12 HOMSTRAD sequences was used as a reference to determine the quality of the alignment produced when the unaligned versions of these sequences were included with the Pfam Cytochrome P450 sequences. Specifically, the proportion of correctly aligned columns out of all aligned columns (TC score) in the 12 reference sequences embedded in the larger data sets was used as a measure of the alignment quality.

For the first analysis, random sets of 1,012 Cytochrome P450 sequences were selected from the 21,001 available in Pfam. These were added to the 12 sequences from HOMSTRAD with known structures to make up 1,024 sequences, a power of 2 being necessary to create a perfectly balanced guide tree. The sequences were randomly shuffled and assigned to the tips of a number of different guide trees, ranging from perfectly balanced through increasing levels of chaining to fully chained. The sequences were aligned with Clustal Omega, Mafft and Muscle using each of these guide trees, and the quality of the resulting alignments measured using their TC score. 100 samples were drawn, each sample was aligned with each of the alignment programs using each of the guide trees, and the results are shown in Figure 2.2. There is a clear and simple trend of increasing accuracy going from the balanced to the completely chained guide trees. For Clustal Omega, the random
Figure 2.2: TC scores for 1,024 Cytochrome P450 (Pfam accession number PF00067) sequences with different guide trees, ranging from perfectly balanced to fully chained (all randomly ordered), for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations). The TC scores obtained with each alignment program's default guide trees are shown on the right for reference. (**P < 0.001, 100 samples).
chained trees produced alignments that were slightly worse than those produced by the default Clustal Omega guide trees. With Mafft and Muscle, the chained trees were considerably better than the default ones but this effect was test-case specific. It is not surprising that a balanced guide tree with randomly-placed sequences did badly, but it is surprising that equally random but perfectly chained trees did so well.

Figure 2.3 plots the lengths of the alignments produced versus the TC scores obtained for the various guide trees. There is a clear trend of shorter alignment lengths and increased TC scores with increasing degree of guide-tree imbalance. It is not clear why this trend exists, and no such trend is visible with the default guide trees.

![Figure 2.3: Alignment lengths versus TC scores for 1,024 Cytochrome P450 sequences aligned with Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations), grouped by degree of imbalance. (100 samples for each guide tree).](image)

To further investigate the effects of chaining on alignment quality, similar experiments were carried out across all of the BAliBASE 3 benchmark test set.

2.3.2 BAliBASE

The BAliBASE structural alignment dataset consists of a number of reference sequence sets, each containing a number of test alignments. For each alignment, all sequences in the test alignment
were randomly shuffled, and balanced and chained guide trees then created. As most test cases have only a relatively small number of sequences, it was not feasible to create guide trees with intermediate levels of chaining. In addition, the “balanced” trees were as close to perfectly balanced as possible given the number of sequences available so, for instance, 12 sequences were allocated to 4 clades of 3 sequences each. The balanced and chained guide trees were again used to order the alignment of the sequences and the quality of the alignments measured using the \texttt{bali\_score} program. The mean quality score was calculated for each family from repeated sampling (the trees have random topologies, so sampling was required) and the results are shown in Figure 2.4.

As before, for all reference sets and alignment programs, random chained trees gave significantly higher quality alignments than random balanced trees. In most scenarios, each alignment program’s default guide trees gave the best quality alignments. The BAliBASE families are quite small, however, with the largest having 142 sequences and, as will be shown below, the improvement in alignment quality with random chained guide trees only became apparent with larger numbers of sequences. Interestingly, both the random balanced and random chained guide trees performed best with the BAliBASE Reference 1 subset V2 (labelled BB12 in Figure 2.4), and also considerably better than the default guide trees. The reason for this is unclear. Reference 1 is made up of phylogenetically equidistant sequences, and subset V2 contains families that have at least 4 equidistant sequences, and in which any two sequences share 20–40% identity. In comparison, Reference 1 subset V1 (BB11) also contain families that have at least 4 equidistance sequences and any two sequences share less than 20% identity, but both the random balanced and random chained TC scores are considerably lower than for BB12. While BB12 has more alignments and sequences than BB11 (45 alignments versus 38 and 411 sequences instead of 265), there are more sequences in all of the other Reference sets so it is unlikely that the higher quality score is due to the number of sequences that were aligned.

### 2.3.3 Larger datasets

The effect of guide tree topology on very large alignments was examined next. In Figure 1.3 Sievers et al. (2013) showed that alignment quality tends to drop off for all progressive alignment methods as the number of sequences increases much beyond approximately one thousand sequences. Here balanced and chained guide tree topologies were compared with the default guide trees for each alignment program for different numbers of sequences ranging from 16 up to over 32,000. The four
Figure 2.4: Average TC scores for BAiLBASE reference sets. The alignments were created with randomly-ordered balanced and chained guide trees. The red line indicated the median TC score for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) using default guide trees. (**P < 0.001, 100 samples).

Different ways of allocating the input sequences to the leaves of the balanced and chained guide trees — random, TSP Minimum, Greedy Neighbour Minimum and TSP Maximum — were used. Given the huge alignments and the need to make replicates, three sequence families representing short, medium and long sequences were selected from HomFam: zinc finger (zf-CCHH, Pfam accession number PF00096, 15 HOMSTRAD 3D structures, average length 23.3 amino acids), short-chain dehydrogenases/reductases (sdr, Pfam accession number PF00106, 13 HOMSTRAD structures, average length 163.2 amino acids), and Cytochrome P450 again (average length 331.5 amino acids). The TC scores for the different topologies and sequence allocations are shown in Figure 2.5, Figure 2.6 and Figure 2.7 respectively. While the actual TC scores are different in each set of results, the overall pattern is the same for all aligners and for the three protein families.

In all cases the quality scores for the default guide trees fell off as the number of sequences increased, as was found in Sievers et al. (2013). The sequence ordering appeared to have an impact with balanced trees, as TSP Minimum and Greedy Neighbour Minimum performed better than random or TSP Maximum ordering, and as good as the default trees in some of the larger alignments. With chained trees, the quality scores declined much more slowly than for either default or balanced
Figure 2.5: TC scores for increasing numbers of zinc finger (Pfam accession number PF00096) sequences for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with (a) default and balanced and (b) default and chained guide trees. Four sets of balanced and chained guide trees were created for each dataset, with different sequence ordering: Random, Travelling Salesman (TSP) Minimum, Greedy Neighbour Minimum and TSP Maximum (100 samples per dataset).
Figure 2.6: TC scores for increasing numbers of short-chain dehydrogenases/reductases (Pfam accession number PF00106) sequences for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with (a) default and balanced and (b) default and chained guide trees. Four sets of balanced and chained guide trees were created for each dataset, with different sequence ordering: Random, Travelling Salesman (TSP) Minimum, Greedy Neighbour Minimum and TSP Maximum (100 samples per dataset, except 25 samples for the largest Clustal Omega datasets).
Figure 2.7: TC scores for increasing numbers of Cytochrome P450 sequences for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with (a) default and balanced and (b) default and chained guide trees. Four sets of balanced and chained guide trees were created for each dataset, with different sequence ordering: Random, Travelling Salesman (TSP) Minimum, Greedy Neighbour Minimum and TSP Maximum (100 samples per dataset, except 25 samples for the largest Clustal Omega datasets).
Figure 2.8: TC scores for short-chain dehydrogenases/reductases sequences for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with default and random chained guide trees. 5% of the sequences in each dataset are Cytochrome P450 sequences (*P < 0.05, **P < 0.01, ***P < 0.001, 100 samples per dataset).

trees. In addition, with the exception of the Clustal Omega alignment of 8,192 Cytochrome P450 sequences, optimising the ordering of the sequences had no impact on the alignment quality. Once the alignment size went beyond a few hundred sequences, across all three alignment programs, random chained guide trees produced higher quality alignments than those produced with the default guide trees.

The results presented in Figures 2.5 to 2.7 are for single-domain proteins. In a further experiment, 5% of the short-chain dehydrogenases/reductases sequences were replaced with Cytochrome P450 sequences to examine the effects of chained guide trees when large numbers of outliers are present in the dataset. These results are shown in Figure 2.8, and while the trends are not as clear as the previous results, the effects of chaining were still apparent for the larger alignments.

2.3.4 Small models

At the other end of the scale, small alignments of 4 sequences were examined. Simple test cases were created with 4 randomly selected and ordered Cytochrome P450 reference sequences with
known structure. Randomly-ordered balanced and chained guide trees were created. The sequences were aligned using these guide trees, and the TC scores calculated for the resulting alignments. The TC scores were higher when aligned with the chained trees than with the balanced ones, as shown in Figure 2.9. While the differences in TC scores were quite small they were nonetheless significant when compared pairwise, even with such small datasets. The only difference between both scenarios was the number of steps where two unaligned sequences were aligned with each other. It was with these alignments of unaligned sequences where alignment differences were most likely to happen, and then be propagated throughout the rest of the alignment. With balanced trees this happened twice, while with chained trees only once.

![Balanced tree](image1)

![Chained tree](image2)

Figure 2.9: TC scores for 4 randomly selected and ordered Cytochrome P450 reference sequences for Clustal Omega, Mafft (FFT−NS−2 algorithm) and Muscle (2 iterations) with balanced and chained guide trees (**\(P < 0.001\), 100 samples).

### 2.3.5 HomFam

Finally, in order to determine whether the results for larger alignments seen above were specific to the HomFam families selected, the tests were repeated across all HomFam families. HomFam protein families are single-domain Pfam families which have at least 5 members with known structures in a HOMSTRAD structural alignment. Given the numbers and size of the families, only random chained trees were compared with the default guide trees from each alignment program. Datasets were limited to 1,024, 2,048 and 4,096 sequences per family and only families with at least 4,096 sequences were included in the experiment. There are 94 HomFam families in total, but only 41 families had a sufficient number of sequences and were included in this test. All reference sequences...
were included in a family’s dataset, with the remainder being selected at random to make up the desired numbers of sequences. Each alignment program was run using both its default guide tree and a random chained tree and the TC scores for the alignments produced were then compared. The results are given in Figure 2.10.

![Figure 2.10: Comparison of TC scores obtained for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with default and randomly chained guide trees for different dataset sizes across all 41 HomFam families that have at least 4,096 sequences. For each family, the TC scores obtained with default and random chained guide trees were compared (significance level $\alpha = 0.01$, 50 samples per family).](image)

In general, as the number of sequences increases there was a corresponding increase in the number of families where the TC score obtained with random chained trees was significantly higher than the default TC scores. While the relative increase in families where chained guide trees gave better TC scores was modest for Clustal Omega, it was seen across all three alignment programs. Interestingly, even with a relatively low significance level ($\alpha$) of 0.01, the results showed few families where there was no significant difference between the default and chained guide trees.

### 2.3.6 Optimal ordering

In the results presented above, random chained guide trees on average gave higher quality results for larger datasets. This begs the question whether a specific ordering of sequences on a chained guide-tree topology would yield higher quality alignments than random ordering. While in theory such an optimal ordering should be possible, determining this ordering is not straightforward. The results in Section 2.3.3 showed that a minimisation or maximisation of the distances between sequences as given in the distance matrix did not generally improve the quality of the alignment
produced with a random ordering of sequences.

In order to determine what factors might have contributed to the improvements in TC score by using random chained guide trees, all sequences in each HomFam family, including reference sequences, were randomly shuffled and then aligned. A number of variables were recorded for the input sequences, reference sequences and the alignments produced by both random chained and default guide trees. The shuffling, alignments and variable capture was repeated 50 times for each HomFam family and guide tree. The variables, described in Table 2.1, were then examined using a Random Forest (Breiman, 2001) to see each variable’s impact on the difference in TC scores obtained with chained and default guide trees.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input sequences:</td>
<td></td>
</tr>
<tr>
<td>AverageLength</td>
<td>Average length of the sequences to be aligned</td>
</tr>
<tr>
<td>NoRefSequences</td>
<td>Number of reference sequences included in the alignment</td>
</tr>
<tr>
<td>NoSequences</td>
<td>Number of sequences (including reference sequences) to be aligned</td>
</tr>
<tr>
<td>Guide trees: metrics are from Shao and Sokal (1990) except MaxDepth</td>
<td></td>
</tr>
<tr>
<td>Colless</td>
<td>[\sum_j T_j \quad j = 1, \ldots, k] where (k) is the number of interior nodes in the tree and (T_j) the absolute difference in the number of terminal nodes (OTUs) under the two branches of node (j). Higher values indicate more imbalanced trees</td>
</tr>
<tr>
<td>InverseMax</td>
<td>[\sum_j (1/M_j) \quad j = 1, \ldots, k-1 \quad (j \neq \text{root})] where the summation is over all interior nodes other than the root of the entire tree, and (M_j) is the maximum number of interior nodes between the terminal nodes (i) and interior node (j). Higher values indicate more balanced trees</td>
</tr>
<tr>
<td>MaxDepth</td>
<td>Maximum depth of the guide tree used in the alignment</td>
</tr>
<tr>
<td>Sackin</td>
<td>[\sum_i N_i \quad i = 1, \ldots, t] where (t) is the number of terminal nodes in the tree and (N_i) the number of interior nodes between terminal node (i) and the root (which is included in the count). Higher values indicate more imbalanced trees</td>
</tr>
<tr>
<td>Shannon</td>
<td>[-\sum_i P_i \log P_i \quad i = 1, \ldots, t] where (t) is the number of terminal nodes, and (P_i) is the probability of reaching terminal node (i) starting at the root. It is calculated as: (P_i = \prod_j [1/(d_j - 1)]) (j = 1, \ldots, N_i) where (d_j) is the degree of interior node (j) and (N_i) is the number of interior nodes between terminal node (i) and the root. This is the Shannon-Wiener information function, and higher values indicate more balanced trees</td>
</tr>
<tr>
<td>Aligned sequences:</td>
<td></td>
</tr>
<tr>
<td>AlignmentLength</td>
<td>Length of the aligned sequences</td>
</tr>
<tr>
<td>AverageId</td>
<td>Average percentage of residues identical between each pair of sequences</td>
</tr>
<tr>
<td>MostDistantSeq</td>
<td>The minimum value of the maximum pairwise identity of each sequences</td>
</tr>
<tr>
<td>MostUnrelatedPair</td>
<td>Lowest percentage of residues identical between all pairs of sequences</td>
</tr>
<tr>
<td>MostRelatedPair</td>
<td>Highest percentage of residues identical between all pairs of sequences</td>
</tr>
</tbody>
</table>

Table 2.1: Description of variables used in Variable Importance Plots shown in Figure 2.11.

The Random Forests produced two separate Variable Importance Plots for the three alignment programs, and these are presented in Figure 2.11. The plots show the percentage increase in Mean
Square Error when each of the variables were randomly assigned values from the test dataset (out-of-bag dataset), and the increase in node purity due to every split on a variable. In the left-hand graphs, if the variable was important then randomly assigning it another valid value would increase the error score of the resulting tree. Hence, more important variables appear at the top of the graphs. The right-hand graphs show the increase in node purity (or decrease in node impurity) for each variable when that variable was used as the basis of a split, averaged over all the trees in the random forest. The goal with each tree in the random forest is to produce subsets of the data which are as homogeneous as possible with respect to the target variable (Breiman et al., 1984). As these were regression random forests, the node impurity was measured using residual sum of squares (RSS), and the descendent nodes gave a lower RSS than their parent node. Again more important variables appear at the top of the graphs. Given the different means of determining importance, it is to be expected that the variables have different rankings in the two plots.

The AlignmentLength from both the default and chained guide trees was deemed an important variable across all three alignment programs, as shown previously in Figure 2.3 for a smaller dataset. Interestingly the AverageLength of the input sequences was ranked the most important variable for Mafft alignments, and only slightly lower for Muscle. On examining the data, this appeared to be primarily related to the two families with the longest sequences having strongly positive differences between the chained TC and default TC scores when aligned using Mafft and Muscle: ace, alpha beta hydrolase, Pfam PF00135, average length 427.8; and ghf1, glycosyl hydrolase family 1, Pfam PF00232, average length 403.8. The AverageId of the chained and default alignments, and also between the reference sequences themselves, were among the most important variables for all alignment programs and for both importance criteria. While these variables may explain the differences in default and chained TC scores, they cannot be used to order the alignment of the sequences as the values were derived from the generated alignments and so are not known in advance. The MostRelatedPair and MostUnrelatedPair were consistently towards the bottom of the plots, but this is understandable as these values referred to just two sequences out of each HomFam family. Of the guide tree measures, the InverseMax and MaxDepth for the default guide trees are the most important variables, and the Shannon value the least important.

Unfortunately, based on this brief analysis, it is not clear what ordering would generate alignments of higher quality than random ordering.
Figure 2.11: Variable Importance Plots for the improvement in TC score between random chained and default guide trees for Clustal Omega, Mafft (FFT-NS-2) and Muscle (2 iterations). The left-hand plots show the increase in error score for random permutation of a variable’s values, and the right-hand plots the increase in node purity when a node is split on that variable. Lower-case variable prefixes are ‘r’: Reference sequences, ‘d’: Default guide tree alignments, and ‘c’: Chained guide tree alignments.
2.4 Complexity and run times

There are some immediate and surprising side-effects from the discovery that simple guide trees do so well on protein structure-based benchmarks. The most obvious is the simplifying effect that chained trees have on the performance of some of the most widely used packages for making large protein alignments. The effects on Maft and Muscle are striking. Chained trees gave an increase in accuracy and a large reduction in computational complexity, especially for large numbers of sequences. No additional iterations were needed and the initial trees could be constructed in trivial amounts of time and memory. Computing times are given in Figure 2.12. With Muscle, once a guide tree was made, the time required to create a multiple sequence alignment was the same regardless of the tree topology. Making alignments in Maft using chained guide trees was slower than using the default or balanced trees, but with chained trees the guide-tree creation step can be skipped so there is more of a trade-off involved. With Clustal Omega, while there was an increase in accuracy it was at the cost of a considerable rise in the time to generate the alignments. The default guide trees in Clustal Omega are usually created using mBed which is fast and has $O(N \log(N))$ complexity, so the saving in time at the guide tree construction phase is modest. Once a guide tree is constructed, the alignment times with chained trees were much longer than with the default or balanced ones. From an examination of the Clustal Omega source code, the increase in complexity came from the way Clustal Omega aligned the Hidden Markov Models as it followed the alignment order defined in the guide tree. As the chained guide tree was far deeper than the default tree, the recalculation of parts of the HMM profiles happened far more often with chained guide trees, resulting in longer run-times.

2.5 Chaining in other situations

It was not possible to test the effects of guide tree topology on Kalign2, as it cannot use an externally-generated guide tree. However, it is one of the few alignment programs, in addition to the ones tested above, that can align very large numbers of sequences. Also Kalign2 scores very well on various benchmark studies that explicitly test the quality of the alignment of large numbers of protein sequences.

Figure 2.13 gives an indication of the degree of chaining in the guide trees used by Kalign2 to align 1,024 sequences from each of the HomFam families that have sufficient sequences. These
Figure 2.12: Benchmark times for the alignment step for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with default, balanced and chained guide trees for shorter (zinc finger, zf-CCHH), medium length (short-chain dehydrogenases/reductases, sdr) and longer (Cytochrome P450, p450) sequences. Also included is the elapsed time for the default guide tree creation and the subsequence alignment. All times were determined using a single thread on an Intel Xeon 2.4GHz CPU with 16GB RAM.

trees ranged from being moderately to extremely chained in topology, especially with short sequence lengths. This seems to derive from the use of the Muth-Manber (Muth and Manber, 1996) alignment metric for quickly measuring the similarity of unaligned sequences. For shorter sequences, this gave a score of either 0 or 1 in many cases. A distance matrix that has mainly 0s and 1s will produce highly chained guide trees. This may in fact be one of the reasons why the alignments from Kalign2 appear to be so good.

The Pfam database (Sonnhammer et al., 1997) consists of collections of protein sequence domains arranged into protein families, with accompanying profile Hidden Markov Models and multiple sequence alignments. It was never a stated aim of the Pfam developers to produce high quality alignments. Instead, the large Pfam alignments are produced by a method that is intended to be simple and straightforward rather than intensive. Essentially the Pfam HMMs are built up using
chained guide trees, as the sequences are added to a growing alignment by aligning them in turn to a HMM derived from a core “seed alignment”. However, what were assumed to be low quality alignments seemed able to produce HMMs for sequence searching that were just as useful as ones from more involved alignments (Griffiths-Jones and Bateman, 2002). Testing the large full alignments in some Pfam families using a benchmark based on protein structures has shown the alignments to be remarkably good (Fox et al., 2016). It is quite possible that the supposedly simplistic algorithm that is used to create the large Pfam alignments is the optimal way to do this, given the time constraints involved in doing this for all protein domains.

2.6 Subsequent work

Following publication of the findings presented above in PNAS in July 2014, Tan et al. (2015) replied disputing the results. For them, random chained guide trees gave poorer quality alignments based on simulation results and phylogenetic benchmarks. Their results, shown in Figure 2.14 were based on two datasets, one generated by simulation and the second on gene families sampled from the Orthologous Matrix (Dessimoz et al., 2005) database.

Tan et al.’s simulated data were generated using ALF (Dalquen et al., 2012), a computer program that can simulate a range of evolutionary processes including amino acid substitution, indels, gene
Figure 2.14: Evaluation of alignments reconstructed with various aligners and guide tree methods. (A) Average true column score over 113 simulated datasets of 1,024 sequences. (B) Average consistency with the National Center for Biotechnology Information taxonomy over 106 sets of 1,024 biological sequences. Note that the real tree is unknown for empirical data. With fully imbalanced trees as input guide tree, Prank failed to reconstruct alignments in 38 empirical data problem instances; results reported in B are thus based on the remaining 68 alignments. Significant difference from fully imbalanced guide trees is indicated with an asterisk (Wilcoxon double-sided test, \( P < 0.001 \)). All data available at dx.doi.org/10.5061/dryad.4r5b8. Reproduced from Tan et al. (2015, E100)

duplication, gene fusion and fission, and speciation. They generated 113 separate datasets, starting with random seed sequences, and simulated evolutionary processes until each dataset contained 1,024 homologous sequences. Each dataset was then aligned using Clustal Omega, Mafft, Muscle and Prank (Löytynoja and Goldman, 2005) — the parameters used with each alignment program
are not provided. Random chained, balanced Travelling-Salesman optimised, the default guide tree, “least-squares distance tree estimated using specialized phylogenetic software” (Tan et al., 2015, E99), and the real evolutionary tree were used as guide trees in these alignments. The quality of each alignment was then assessed by comparing the aligned sequences to those generated by the simulation. As shown in Panel A of Figure 2.14, random chained guide trees gave the lowest column scores of any of the guide trees across all alignment programs examined.

In addition, Tan et al. aligned 1,024 homologous sequences extracted from the Orthologous Matrix database, aligned these sequences using a random chained, balanced optimised, the alignment program’s default guide tree, or a guide tree inferred from an alignment using “specialized phylogenetic software”. From the alignments produced, phylogenetic trees were inferred and their congruence compared with the National Centre for Biotechnology Information taxonomy. Their assumption was that more accurate alignments should yield more accurate phylogenetic trees, which then should have a higher relative congruence score with the NCBI taxonomy, and they argued that there is a clear correlation between the accuracy of the guide tree and the resulting phylogenetic tree. Their results are shown in Panel B of Figure 2.14, and again random chained guide trees gave the lowest scores across each of the four alignment programs.

Examining the results in more detail, it is curious that the default guide trees for Mafft, Muscle and Prank resulted in higher congruence scores than the alignments produced with a guide tree inferred by the specialised software, and the default guide tree in Clustal Omega gave only a marginally lower congruence score. As will be shown in Chapter 3, the guide trees for Clustal Omega, Mafft and Muscle were determined by the number of shared $k$-tuples (1-tuples for Clustal Omega, and 6-tuples for Mafft and Muscle). It is not mentioned in the Prank paper (Löytynoja and Goldman, 2005) how its guide trees are constructed. Such approaches only score exact matches between sequences. Weakly similar sequences are considered to be completely unrelated. Hence, the Clustal Omega, Mafft and Muscle guide trees should be seen as part of the heuristic progressive multiple sequence alignment approach, and not as versions of the true phylogenetic trees for these sequences.

We were given the opportunity of replying to Tan et al., and both letters were published at the same time. Given the tight time and space constraints for such replies, we focused on the results using the simulated sequences that is presented in Panel A of Figure 2.14.

Firstly, it should be noted that the calculation of the quality or column score differed in Tan et al.
and in our original manuscript. In Tan et al., the Column Score was given by the number of correct columns \textit{including gaps} divided by the number of columns in the reference or “true” alignment. In the figures presented earlier, the TC score was Edgar’s implementation of the Thompson et al. (2005) Column Score. This calculation ignores residues in the alignment in positions where there are gaps in the reference sequences. To illustrate this, consider the following simple test and reference sequences:

<table>
<thead>
<tr>
<th>Test</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;SEQ1</td>
<td>&gt;SEQ1</td>
</tr>
<tr>
<td>ACD</td>
<td>ACD -</td>
</tr>
<tr>
<td>&gt;SEQ2</td>
<td>&gt;SEQ2</td>
</tr>
<tr>
<td>A-E</td>
<td>A--E</td>
</tr>
</tbody>
</table>

For Tan et al. this alignment has a Column Score of 0.50, as only the first two columns of the four in the reference sequences were matched in the test sequences. With Thompson et al.’s method however, the alignment would get a TC score of 1.00 as all columns were correctly matched: a column with two ‘A’s, and columns with a ‘C’ and something else, a ‘D’ and something else and an ‘E’ and something else.

In addition, Tan et al.’s simulation stopped at 1,024 sequences, whereas the improvement in quality due to chaining reported earlier was most clearly visible for larger dataset sizes. In order to determine if this was a reason for the differences in the results, Tan et al.’s simulation was continued to 4,096 sequences, and the simulated sequences were aligned using both a default and random chained guide tree using Muscle. The quality scores were then calculated using both Edgar’s \texttt{qscore} program and the Darwin code used by Tan et al.. The results are shown in Figure 2.15 and, as can be seen, random chained guide trees produced lower scores in all cases. However, using the existing simulation parameters the \texttt{qscore} TC score actually increased as the number of sequences being aligned increases, contrary to the results shown in Figure 1.3 where the TC score declined with increasing number of sequences for all alignment programs. Reducing the indel rate to one-fifth of its value reverses this trend.

Our fundamental objection, however, was that Tan et al.’s results were based on simulated sequences. While simulations can yield insights into difficult-to-observe phenomena, implausible conditions or parameter values can yield unrealistic results. When simulating sequence evolution, there...
are a number of parameters that must be specified for modelling indels: where they can occur in the sequence, how likely they are to occur and how long the indels will be if they do occur. Tan et al. chose a uniform distribution of indels across the protein sequences, even though it has been known for some time that gaps are typically confined to loops between the main secondary structure regions (Pascarella and Argos, 1992). The article announcing ALF (Dalquen et al., 2012), when describing the indel length distributions that can be used, references a paper by Chang and Benner (2004) showing that indel lengths tend to follow a Zipfian distribution — Tan et al. specify such a distribution for indel length in their simulation. However, Chang and Benner also describe how breaks tend to occur at exposed surfaces of protein structures and not with equal likelihood throughout the protein. A uniform probability of indels occurring at all locations in a protein sequence would suggest that the protein is not under any selective or structural constraint. The tests presented earlier in this chapter, on the other hand, used known Pfam sequences and HOMSTRAD structure-based alignments.

The underlying difference between the two sets of sequences can be seen in Figure 2.16, where the data are differentiated based on the average pairwise identity between the aligned sequences and the increase in length of the aligned sequences over the unaligned ones. The simulated se-
Figure 2.16: The length of the longest sequence in a protein family is given along the x axis; length of the final alignment is along the y axis. The alignments were produced using the phylogeny-aware program PAGAN. The 41 HomFam datasets (HF), as used in figure 5 of Boyce et al. (2014), are rendered as solid squares; the 200 simulated datasets (Sim), as used in Tan et al. (2015), are shown as open circles. The average pairwise identity in the alignments is rendered with color (blue/green, high identity; red/yellow, low identity). Lines represent average “inflation” of the alignments because of the alignment process: solid line 40-fold inflation for HomFam, dotted line 12-fold inflation for the simulated data.

sequences have a much higher average pairwise identity than the HomFam sequences. In addition, while the unaligned HomFam and simulated sequences have similar lengths, aligning the datasets with PAGAN (Löytynoja et al., 2012) results in much greater alignment lengths for the HomFam sequences than for the simulated sequences. These differences in the nature and behaviour of the two sets of sequences calls into question any direct comparison of the alignments produced from them.

2.7 Discussion

The results presented in Section 2.3 above indicate that, on average, for alignments of hundreds or thousands of single-domain protein sequences, chained guide trees give higher quality results
than default or balanced guide trees when the alignment quality is measured using structure-based benchmarks.

This has some clear implications for both the developers and users of multiple sequence alignment programs or databases: To make alignments of very large numbers of protein sequences using a progressive alignment method, the optimum solution may be to simply add the sequences one at a time, in any order, to a growing alignment. As shown previously, this can give a clear and immediate jump in accuracy across three of the main alignment programs. In addition, the guide trees used in these alignments are trivial to create and no alignment iterations are needed to refine their topology. Skipping the existing guide-tree building procedures also results in faster runtimes for Muscle, although the net effect on Mafft runtimes is neutral and significantly worse for Clustal than using the default guide trees.

A key question remains as to why chained trees do so well when aligning large numbers of sequences. The simple four sequence example shown in Figure 2.9 hints at a possible explanation. In that, there is a small but significant improvement in accuracy using random chained versus balanced guide trees. This effect is amplified when the alignment is scaled up to hundreds of sequences. With fully-chained guide trees, pairs of unaligned sequences are only aligned once at the start of the alignment. After this, all other alignments involve aligning a sequence against a profile of already-aligned sequences. With default guide trees, however, the alignment of pairs of unaligned sequences occurs far more often, depending on the number of sequences involved and how balanced the topology of the default guide tree. As each alignment step is an attempt to find an optimum match between the items being aligned, aligning many pairs of unaligned sequences may result in a poorer overall alignment as these pairs of sequences are subsequently aligned into larger groups. Hence, in Figure 2.3, random balanced guide trees generate longer alignments with more gaps than random chained trees. With chained guide trees, after the first pair of sequences are aligned, the alignment of all other sequences are constrained by the growing overall alignment. This is essentially what is happening in Figure 2.9, as the third and fourth sequence in the chained guide tree are aligned to the innermost pair of sequences in the tree.

It is acknowledged that the main results presented in this chapter may not hold up when viewed from a strictly phylogenetic perspective, or if the main aim is to infer the precise positions of gaps in the alignment (e.g. Löytynoja and Goldman (2005)). It goes without saying that the chained guide trees proposed in this chapter are intended solely for aligning the given sequences, and are not
meant as phylogenetic trees. As Shao and Sokal (1990, p268) point out, a chained phylogenetic tree is unlikely to occur, as “one branch at each furcation should have a high probability of speciation soon after forming, followed by a low speciation probability after forming the new lineage”. There is a problem in the field when trying to reconcile the apparently conflicting results from benchmarks based on evolutionary models and simulations, versus those based on 3D protein structures of proteins (Iantorno et al., 2014). This issue is beyond the scope of this chapter, and remains to be resolved.
Chapter 3

Instability in Alignment Algorithms


3.1 Introduction

As described in Chapter 1, creating a multiple sequence alignment is a routine step in the analysis of homologous genes or proteins. The standard method for aligning more than a few hundred sequences is termed “progressive alignment” by Feng and Doolittle (1987): a guide tree (Higgins et al., 1992) is first created by clustering the sequences based on some distance or similarity measure, and then the branching structure of the guide tree is used to order the pairwise alignment of sequences. Progressive multiple sequence alignment is popular because, as a heuristic approach, it enables the alignments of thousands of sequences on mid-level desktop computers within minutes or a few hours. The power of progressive multiple sequence alignment may come from the fact that “more similar” sequences are aligned first “...assuming that in progressive alignment, the best accuracy is obtained at each node by aligning the two profiles that have fewest differences, even if they are not evolutionary neighbours” (Edgar, 2004b, p1792).

To construct a guide tree, all sequences are compared to each other to generate a matrix of distance measures between each sequence pair. The calculation of these distance measures should be fast as it requires $O(N^2)$ time and memory for $N$ sequences. Most alignment programs use $k$-tuple scores (Dumas and Ninio, 1982; Wilbur and Lipman, 1983), or comparable word-based measures, to determine the similarity of two sequences. Some programs use other string-matching
algorithms to the same effect. While these approaches are fast, there is an associated loss of precision as they only score exact matches between two sequences. For proteins, amino acids that are different but considered very similar according to, say the PAM (Dayhoff and Schwartz, 1978) or BLOSUM (Henikoff and Henikoff, 1992) matrices, are treated as complete mismatches. Also, the position of the matches within the protein sequences is often ignored.

This chapter examines the impact of this tradeoff of accuracy for speed in the construction of the guide trees in protein progressive multiple sequence alignment. The results show that, because of a loss of information when calculating the distance measures, the alignments generated are inherently unstable. Simply changing the order of the protein sequences in the input file will often cause a different alignment to be generated. In addition, while this instability is more apparent with larger alignments and with some alignment programs, it is also found in small alignments of less than 100 sequences.

This instability is due to huge numbers of tied scores in the distance matrices used to make the guide trees. With word-based distance measures, there is a relatively small number of possible distance scores that can be found between two sequences. This number will depend on the length of the sequences involved and on the specific metric used. Hence, once a certain number of sequences is reached, the distance matrix will have many tied scores, which ideally would be represented as multifurcations in the guide tree. Progressive alignment uses bifurcated guide trees, and the branching order selected from these tied groups is completely arbitrary and determined purely by how the clustering code was written. If the sequence order is changed, the clustering order will also be changed, and so too will the progressive alignment order specified by the guide tree. This means that the supposed power of the guide tree to sensibly align the sequences in the correct order is lost and the considerable computation effort required for their calculation may be completely wasted.

In addition, the reproducibility of any analysis based on a multiple sequence alignment may be in question. While most research findings will define the materials and methods used so that others may reproduce results and build upon them, very few will specify the sequence order used in the multiple sequence alignment. Without knowing this ordering, a different alignment may be produced leading to conflicting or inconclusive results in downstream analyses.

It should be noted that the instability described below is different from the Heads or Tails reliability check proposed by Landan and Graur (2007). This relies on the assumption that biological
sequences do not have a particular direction, so the same alignment should be produced by a set of sequences and the same sequences when the residues are reversed within each individual sequence. The agreement between the two alignments can then be determined and presented as a measure of the reliability of the alignment. The underlying reason for any differences in the alignments of the original and reversed sequences is due to equally optimal paths through the dynamic programming matrix when deciding to insert a gap in one sequence or the other. The two options were termed the “High Road” and the “Low Road” by States and Boguski (1991), and will result in different alignments being produced depending on which path is chosen. Unfortunately, the choice of which path to take is purely arbitrary, and only depends on how the programmer specified the condition within the alignment program. Reversing the order of the residues within each sequence will result in the other “road” being taken if it is equally optimal.

While both the instability described in this chapter and the High-Road / Low-Road variability are both due to the specification of conditions in the alignment programs, they happen at different stages within the progressive alignment process. The choice to take either the high road or the low road when inserting a gap within a sequence occurs towards the end of the process when aligning two sequences or profiles. The focus of this chapter occurs much earlier, during the clustering phase when the guide tree is being constructed and before the actual alignment of sequences has commenced.

3.2 Methods

3.2.1 HomFam

The analysis presented here used the HomFam alignment benchmark system (Blackshields et al., 2010). This consists of the single-domain Pfam (Sonnhammer et al., 1997, version 25) families which have at least 5 members with known structures in a HOMSTRAD (Mizuguchi et al., 1998b) structural alignment. Alignment quality was measured as the proportion of correctly aligned core columns out of all aligned core columns in the reference sequences when these sequences were embedded in larger datasets (Thompson et al., 1999, BAliSCORE TC score). TC scores range from 0.0 (no core columns in the reference sequences correctly aligned) to 1.0 (all reference sequence core columns correctly aligned). It is also possible to calculate the TC score as the proportion of all correctly-aligned columns. While the results are similar when using all columns, the results given
below are for core columns.

On examining the HomFam sequences, it was noticed that a number of proteins had the same amino acid sequence even though they were (correctly) labelled differently in Pfam. As an example, in the zinc finger family (Pfam accession number PF00096), the sequence information for:

```
>D2I3U5_AILME/95-116
ACADCGKTFSQSSHLVQHRRIH
```

and

```
>ZN787_HUMAN/95-116
ACADCGKTFSQSSHLVQHRRIH
```

are identical. Table 3.1 shows the number of sequences in each HomFam family and the number of these that are unique. In the remaining analysis, duplicate sequences have been removed from the HomFam families. The presence of duplicate sequences would have given tied distance measures, and that would have impacted on the $k$-tuple scores obtained.

### 3.2.2 Software

This research investigated the instability of the alignments produced by the progressive multiple sequence alignment programs Clustal Omega (Sievers et al., 2011), Kalign (Lassmann et al., 2009), Mafft (Katoh et al., 2002) and Muscle (Edgar, 2004b). These programs were selected based on their widespread use, their ability to align more than a thousand protein sequences, and their use of a guide tree based on the similarity between each pair of sequences to determine the order in which the sequences are aligned.

Each of the alignment programs generates a distance matrix containing the similarity or distance measures between all pairwise combinations of input sequences. Kalign does not output this distance matrix by default, but on examining its source code, in `kalign2_main.c` line 135 the code to output the distance matrix has been commented out. This code was uncommented and modified to output the distance matrix to a specific text file. In addition, the distance measures were output to 25 decimal places to ensure that any duplicate measures were not as a result of rounding when formatting the output.

The other three alignment programs were also modified to output distance measures to 25 decimal places: Clustal Omega: line 327 of `clustal/symmatrix.c`; Mafft: line 2643 of `io.c`; and
Table 3.1: Duplicate sequence percentages in HomFam protein families. The list of HomFam protein families, the total number of sequences in each family, the number of unique sequences, and the percentage of the total number of sequences that are duplicates.
Muscle: line 59 of fastclust.cpp.

For all four alignment programs, the runtime parameters were limited to those required to generate a distance matrix. By default, Clustal Omega uses the mBed algorithm (Blackshields et al., 2010) to cluster the sequences on the basis of a small number of “seed” sequences. This only requires the calculation of the similarity measures between these seed sequences and all other sequences in the input file. Instead of using this default, the generation and output of a full distance matrix was specified, so that the sequences were clustered using the similarity measures between all pairs of input sequences.

For Mafft, the FFT-NS-1, FFT-NS-2 and G-INS-1 algorithms were used. With FFT-NS-1, a distance matrix is first generated using the 6-tuple score between each pair of sequences — both sequences are scanned from the start for matching 6-tuples, and when a match is found the score is incremented and scanning continues from the next residue (Dumas and Ninio, 1982). A guide tree is then constructed by clustering according to these distances, and the sequences are then aligned using the branching order of the guide tree. With FFT-NS-2, the alignment produced by the FFT-NS-1 method is used to regenerate the distance matrix and the guide tree, and then a second progressive alignment step is carried out using the revised guide tree. In this study, FFT-NS-1 was specified whenever distance measures were needed. If no distance measures were required, the default FFT-NS-2 method was used. The G-INS-1 algorithm, which uses Needleman-Wunsch (Needleman and Wunsch, 1970) pairwise alignment to calculate the distance between sequence pairs, was also used in Figure 3.1 for comparison with a distance measure that doesn’t rely on matching $k$-tuples.

With Muscle, the number of iterations was limited to 2 rather than the default of 16. This is the number of iterations recommended by the program’s author for large datasets.

The program versions and runtime parameters used are as follows:

**Clustal Omega** (v1.2.0-r289):

```
-i <sequence file> -o <alignment file> --full
--distmat-out=<distance matrix file>
```

**Kalign** (v2.04):

```
-i <sequence file> -o <alignment file> -q
```

**Mafft FFT-NS-1** (v7.029b):

```
--retree 1 --anysymbol --distout <sequence file> > <alignment file>
```
Mafft FFT-NS-2 (v7.029b):
   --anysymbol --distout <sequence file> > <alignment file>

Mafft G-INS-1 (v7.029b):
   --anysymbol --globalpair <sequence file> > <alignment file>

Muscle (v3.8.31):
   -in <sequence file> -out <alignment file> -maxiter 2
   -DistMx1 <distance matrix file>

3.2.3 Supporting material

Details of utility programs and scripts used are given in Appendix A.

3.3 Results

In the following sections, reference is made to distance matrices and the calculation of distances between sequences. Most of the cases actually use similarity scores. Nonetheless these can be easily converted to distances and the use of the words distance and distances is retained out of convenience.

3.3.1 Alignment instability

For each of the 94 HomFam families, a random selection of sequences was made so that, when combined with the family’s HOMSTRAD reference sequences, there were 1,000 sequences in total. Families with an insufficient number of sequences were excluded, leaving a total of 68 families.

The 1,000 sequences were randomly shuffled, a default alignment was generated (for Mafft, both the FFT-NS-2 and G-INS-1 algorithms were used), and the alignment quality measured using the BAliSCORE TC score for its core columns. The order of the sequences in the input file was then reversed (the first sequence was listed last, the second sequence second last, etc.), the alignment repeated with the same parameters and the quality of the new alignment measured. The difference between the two quality scores was then calculated. A total of 10 samples were randomly selected for each of the 68 HomFam families, and this process was repeated with each of them. The results are presented in Figure 3.1.

In the first four panels, and for virtually all of the represented HomFam families, reversing the
Figure 3.1: The difference in the TC scores calculated on the core columns for 1,000 randomly-selected sequences and in reverse order. 68 HomFam protein families. $n = 10$ samples per family.
order in which the sequences are listed in the input file had an impact on the quality of the alignments produced. For some protein families and alignment programs this impact was considerable, with the alignment of up to 50% of core columns in the reference sequences changed by reversing the order of the input sequences. In the fifth panel, Mafft G-INS-1, although some instability was still present, it was significantly lower than for the other alignment programs using their default parameters.

It should be noted that Mafft's G-INS-1 was considerably slower than FFT-NS-2 for the given number of sequences, and took approximately two orders of magnitude longer to run. It also required over ten times more memory, and both memory and time requirements scaled quadratically. For these reasons, it is not recommended for aligning more than a few hundred sequences, but was included in the figure for reference purposes. In the remainder of this analysis, only the standard distance measure calculations used when aligning larger numbers of sequences was examined.

### 3.3.2 Unique distances

Clustal Omega uses 1-tuple scores to determine the distance measures between proteins, where the scores are calculated in the same way as Mafft’s 6-tuple score except for the different lengths of matching string. Muscle uses 6-tuple scores calculated in the same way as Mafft, and Kalign uses the Muth-Manber (Muth and Manber, 1996) approximate string matching algorithm. Such methods essentially count the number of matches between sequences, ignoring both the position of the matches and the actual values matched. The number of matches between sequences is therefore related to the lengths of the sequences. Highly divergent, unrelated pairs of sequences can generate the same distance measure as two more closely related sequences. In addition, the chances of seeing such matches will increase as the number of sequences being aligned increases. It is not clear how the clustering algorithm used in each of the alignment programs resolves such ties in distance measures. However, unless this scenario is specifically catered for, the default approach will be to choose between pairs of sequences based on their positions in the input file, usually either the first pair with that distance measure or the last pair.

In order to further investigate the frequency of such tied distances, the number of unique distance values in a distance matrix for different datasets was determined. The four alignment programs, Clustal Omega, Kalign, Mafft (FFT-NS-1) and Muscle were run with the parameters listed previously on random samples of sequences drawn from each of the HomFam protein families. Sample sizes ranged from 50 to 10,000 sequences (or as many unique sequences as were in the HomFam family),
and each sampling was repeated 100 times. As quality scores were not required, the HOMSTRAD reference sequences were not included in each sample. The number of unique distances were counted in the distance matrices produced by each alignment program, and the mean number of unique distances for each family and number of sequences was determined, and are presented in Figure 3.2.

The number of unique distances generated by Mafft was considerably higher than for the other alignment programs. However, for all alignment programs, the numbers of unique distances showed clear trends of levelling off as the number of sequences increases. In addition, as the total number of distances calculated is given by $N(N - 1)/2$ for $N$ sequences, for the larger data sets the vast majority of distance measures were duplicated in each alignment program. Even with Mafft, for the largest alignments, over 96% of entries in the distance matrix have tied values.

### 3.3.3 Distance measure calculations

To determine why the number of unique distances approached a plateau while the total number of pairwise distances increases quadratically, the calculations of the distance measures between sequences were examined.

As mentioned above, Clustal Omega uses 1-tuple scores for comparing sequences. With sequences of the same length it can therefore generate a maximum of $L + 1$ 1-tuple scores, where $L$ is the length of the sequences. These values correspond to no matches, 1 match, 2 matches, etc. up to identical sequences. The 1-tuple scores are then scaled by the length of the shorter of the two sequences, and subtracted from 1.0 to give distance scores that range from 0.0 (identical) to 1.0 (complete difference).

To demonstrate this, the following simple test sequences:

```
>SEQ0
ACDE
>SEQ1
AAAA
>SEQ2
CCCC
>SEQ3
```
Figure 3.2: The number of unique distances with increasing number of sequences. Each line is the mean of 100 samples for a HomFam protein family.
generated this distance matrix (for readability, distance measures in the text have been rounded to four decimal places):

```
5
SEQ0  0.0000  0.7500  0.7500  0.7500  0.7500
SEQ1  0.7500  0.0000  1.0000  1.0000  1.0000
SEQ2  0.7500  1.0000  0.0000  1.0000  1.0000
SEQ3  0.7500  1.0000  1.0000  0.0000  1.0000
SEQ4  0.7500  1.0000  1.0000  1.0000  0.0000
```

Mafft and Muscle use 6-tuple scores, so the maximum number of matches between sequences of length \( L \) is \((L - 5) + 1\) where \((L - 5)\) is the number of 6-tuples in a sequence and the additional +1 is necessary if no matches are found. Muscle's similarity scores are scaled by the number of 6-tuples in the shorter of the two sequences, and adjusted to range from 0.0 (identical) to 3.0 (complete difference). The following sequences:

```
>SEQ0
DEFGHIK
>SEQ1
DEFGHH
>SEQ2
DEFGHI
>SEQ3
DEFGHIK
>SEQ4
DEFGHIL
```
produced this distance matrix:

<table>
<thead>
<tr>
<th></th>
<th>SEQ0</th>
<th>SEQ1</th>
<th>SEQ2</th>
<th>SEQ3</th>
<th>SEQ4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ0</td>
<td></td>
<td>3.0000</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ1</td>
<td>3.0000</td>
<td></td>
<td>3.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ2</td>
<td>0.0000</td>
<td>3.0000</td>
<td></td>
<td>0.0000</td>
<td></td>
</tr>
<tr>
<td>SEQ3</td>
<td>0.0000</td>
<td>3.0000</td>
<td>0.0000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ4</td>
<td>1.5000</td>
<td>3.0000</td>
<td>0.0000</td>
<td>1.5000</td>
<td></td>
</tr>
</tbody>
</table>

In Mafft, the distance measure is calculated as:

\[
D'_{ij} = \frac{D_{ij}}{f(x, y)}
\]

where:

\[
D_{ij} = 1 - \frac{S_{ij}}{\min(S_{ii}, S_{jj})}
\]

\[
f(x, y) = \frac{y}{x} \times 0.1 + \frac{10000}{(x + 10000)} + 0.01
\]

\(S_{ij}\) is the 6-tuple score between sequences \(i\) and \(j\), and \(x\) and \(y\) are the lengths of the longer and shorter sequences respectively. The additional scaling is deemed necessary as \(D_{ij}\) can be close to zero when comparing very short and very long sequences, even if the sequences are unrelated.

The sample sequences used for Muscle above resulted in this distance matrix:

<table>
<thead>
<tr>
<th></th>
<th>SEQ0</th>
<th>SEQ1</th>
<th>SEQ2</th>
<th>SEQ3</th>
<th>SEQ4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEQ0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ1</td>
<td>2.306</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SEQ4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1

5

2.306

1. =SEQ0
2. =SEQ1
3. =SEQ2
4. =SEQ3
5. =SEQ4
The Kalign distance calculations are somewhat more complex, in part because the Muth-Manber algorithm allows for matches between sequences with one error. In addition, patterns seemingly must be significantly longer than the 6-tuples in Mafft or Muscle before matches are recognised. As an example, these 12-residue long test sequences:

>SEQ0
DDDDDDDDDDDD
>SEQ1
DDDDDDDDDDDE
>SEQ2
DDDDDDDDDDDEE
>SEQ3
EEEEEEEEEEEE

produced a zero-filled distance matrix:

\[
\begin{array}{cccc}
0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 \\
0.0000 & 0.0000 & 0.0000 & 0.0000 \\
\end{array}
\]

However, when the sequences were increased by 1 amino acid:

>SEQ0
DDDDDDDDDDDD
>SEQ1
this distance matrix was generated:

```
  0.0000 124.0000  0.0000  0.0000
124.0000  0.0000  59.0000  0.0000
  0.0000  59.0000  0.0000  0.0000
  0.0000  0.0000  0.0000  0.0000
```

Curiously, if the run of the repeated amino acid was broken, as in these sequences:

```
>SEQ0
DEDDDDDDDDDDD

>SEQ1
DEDDDDDDDDDE

>SEQ2
DEDDDDDDDEE

>SEQ3
EDEEEEEEEEE
```

the all-zeros distance matrix was produced. The same all-zeros distance matrix was also produced with these 31-amino acid long sequences:

```
>SEQ0
DEDEDEDEDEDEDEDEDEDEDEDE

>SEQ1
DEDEDEDEDEDEDEDEDEDEDEED
```
Hence, the length of the pattern being matched appears to depend on the contents of that pattern.

Regardless of whether this was an intended feature or not, Kalign produced far fewer unique distances than the other alignment programs. Further comments on the Kalign distance measure calculations are given below in Section 3.3.6.

All of these distance measure calculations ignored both the position of the matches and the actual values matched, thereby greatly reducing the information content available when comparing the sequences. It is this loss of information that results in so many tied distance measures, with the resulting alignment instability.

### 3.3.4 Theoretical maximum number of unique distances and sequences

Based on this analysis, the two factors that determine the number of different possible distance measures are the lengths of the sequences being aligned and the number of different sequence lengths included in the input file. For simplicity, the minor adjustments to the sequence lengths due to using 1-tuples or 6-tuples will be ignored. Hence, for Clustal Omega, Kalign and Muscle, the theoretical maximum number of unique distances is given as the product of the longest sequence length and the number of different sequence lengths in the dataset. For Mafft, as both sequence lengths are included in the additional scaling factor, the theoretical maximum is the longest sequence length times the square of the number of different sequence lengths. These theoretical maxima are conservative as all sequences may not be as long as the longest sequence, and all possible matches for all sequence lengths may not be found. So, for Clustal Omega, Kalign and Muscle:

\[
\text{MaxUniqueDists} = \text{MaxSeqLength} \times \text{Count(SeqLengths)}
\]

and for Mafft:

\[
\text{MaxUniqueDists} = \text{MaxSeqLength} \times \text{Count(SeqLengths)}^2
\]
where $MaxUniqueDists$ is the theoretical maximum number of unique distances, $MaxSeqLength$ is the length of the longest sequence in the dataset, and $Count(SeqLengths)$ is the number of different sequence lengths. In addition

$$MaxSeqs(MaxSeqs - 1)/2 = MaxUniqueDists$$

where $MaxSeqs$ is the maximum number of sequences that can be aligned before duplicate distance measures are generated.

Figure 3.3 plots these theoretical maxima for Clustal Omega, Kalign and Muscle (Panel a), and Mafft (Panel b) for each HomFam family based on all sequences in each family. Also shown are the maximum numbers of unique distances for each family found in the datasets used to construct Figure 3.2 previously. As can be seen, for each of the alignment programs the pattern of unique distances in the datasets follows but is lower than the theoretical maxima.

The lower plots in Figure 3.3 show the maximum number of sequences for each family that can be aligned without duplicate distance measures, derived from these maximum numbers of unique distances. Again these maximum numbers of sequences are a conservative measure, as they are based on all lengths of sequences occurring in the dataset and each sequence having its full range of possible matches. Perhaps the most striking thing about the lower plot is that the numbers of sequences are so low, particularly for Clustal Omega, Kalign and Muscle.

It should be noted, however, that duplicate distance measures do not necessarily lead to instability in the alignment generated. Alignment instability will depend on whether the duplicate measures are the lowest values in the distance matrix at a particular step in the clustering process, which will in turn depend on what has happened in the previous clustering steps. Hence, it cannot be said with certainty that duplicate measures will lead to alignment instability. However, as the number of duplicate measures increases, so too does the likelihood of alignment instability.

3.3.5 Smaller alignments

While the instability demonstrated earlier was more apparent in larger alignments, it may also be present when smaller numbers of sequences are aligned. This can be shown by randomly selecting 50, 100 and 250 sequences (including each family’s reference sequences) from each HomFam family that has a sufficient number of sequences, and calculating the TC scores for the forward and
Figure 3.3: The theoretical maximum number of unique distances for each HomFam family, the actual number of unique distances found in the datasets used to generate Figure 3.2, and the maximum theoretical number of sequences that can be aligned without generating duplicate distance measures based on the calculation that $N$ sequences will produce $N(N-1)/2$ distance measures. Left-hand panels Clustal Omega, Kalign and Muscle (2 iterations). Right-hand panels Mafft (FFT-NS-1). The protein families are ordered according to theoretical maximum number of unique distances along the horizontal axis.
reversed datasets, as was done in Figure 3.1. 100 random samples were used for each HomFam family and for each of the three dataset sizes. For each sample, the forward and reverse TC scores were compared, and the number of differences for each HomFam family were counted. These counts are shown in Figure 3.4.

![Graph showing number of unique distances vs number of different forward and reverse TC (core) scores for different alignment programs and dataset sizes.]

Figure 3.4: The number of samples within each HomFam family where the forward and reverse TC core scores are different. \( n = 100 \) samples for each family and dataset size.

As can be seen, the instability in sequence alignments occurs even with small alignments. Also, as the number of sequences increases so too does the number of differences in TC scores. While there is no clear trend between the number of unique distances and the number of TC score differences for a particular alignment program, which might be expected, this trend can be seen across the different programs — Mafft shows the fewest number of differences in TC scores and Kalign the most.
3.3.6 Algorithm symmetry

Another reason for the difference in TC scores reported above may be due to the asymmetry of the different implementations of distance measures. Different distance measures could then cause a different clustering order and give a different tree topology, causing sequences to be aligned in a different order.

To illustrate, two Retroviral aspartyl protease (Pfam accession number PF00077) sequences were randomly selected, aligned using the four different alignment programs and the distance measures between the two sequences extracted. The order of the two sequences was then reversed, the alignment programs run again, and the distance measure from this second run captured and compared with the original. (Clustal Omega requires a minimum of three sequences, so three sequences were selected at random and the distances between the first and third sequences were compared.)

Out of 10,000 samples, for Clustal Omega there were 9 different distances identified. With Mafft and Muscle, no different distance measures were found. However, with Kalign 6,516 differences were found. It is not clear whether these differences were intended by the authors of the alignment programs, or whether they are due to code defects. Regardless of the rationale, the generation of the distance measures will clearly depend on the input sequence ordering, and hence too will the guide trees and the alignments produced, if the calculation of the distance between two sequences is not symmetric.

3.4 Conflicting instability

In discussions with Cedric Notredame and Maria Chatzou of the Centre For Genomic Regulation, Barcelona, they pointed out that the High-Road / Low-Road variability can also be a source of instability based on the ordering of the sequences in the input file, as the ordering of the leaves under the lowest bifurcation of a guide tree is determined by the ordering of the sequences in the input file. In addition, the insertion of gaps in one sequence or another at the lowest level of a guide tree may have a knock-on effect in alignments further up the guide tree. Hence, a different ordering of the input sequences may lead to a different alignment being produced, even with the same guide tree being used in both alignments.

To determine the relative impact of both of these sources of instability, random samples were
taken from each of the 68 HomFam families that have at least 1,000 sequences (including HOM-STRAD reference sequences). The sample sizes were 50, 100, 250 and 1,000 sequences, again including reference sequences, and 100 samples of each size were taken. The positions of the sequences in each sample were randomised and a default alignment created using Clustal Omega, Kalign, Mafft and Muscle — for comparison with Figure 3.4, the FFT-NS-1 algorithm was used in Mafft. The TC core column score was measured for each alignment, and the guide tree generated was saved. The positions of the sequences in the input files were then reversed, the alignment programs were run again with the same parameters, and the TC core column scores and guide trees recorded. The Robinson-Foulds distance (Robinson and Foulds, 1981) between the forward and reverse guide trees for each sample was calculated using the treedist program from the phylip package. Figure 3.5 shows the number of differences in either the guide trees or TC core scores due to reversing the position of the input sequences. Cases where there are differences in the guide trees are due to the tied distance measures described in earlier sections, and those cases with differences in TC scores with the same guide trees are due to the High-Road / Low-Road instability.

As can be seen, with the exception of small alignments generated with Mafft, the majority of alignments result in a different guide tree being created when the sequence order is reversed, and this effect becomes more pronounced as the numbers of sequences is increased. The number of cases of ‘No instability’ is highest for Mafft, which is expected given its use of both sequence lengths when calculating the distances between sequence pairs. The differentiation between having the same or different TC scores when a different guide tree is used is as much down to chance as anything else, as the core and non-core residues are treated the same during the alignment step. It is also possible that different alignments give the same TC score due to offsetting changes in different columns of the alignments. Of more relevance to the current topic, however, is the low number of cases where the same guide tree was created, but a different TC score was generated. The use of the TC score for just core columns possibly understates the full extent of High-Road / Low-Road instability, which also applies to the ‘Different tree, different TC score’ cases.

3.5 Discussion

The results above demonstrate a very strong dependence on the order of the input sequences in a data file when measuring multiple sequence alignment accuracy. This effect is disconcerting as merely changing the order of the sequences can change the alignment produced. The scale of this
Figure 3.5: The number of samples within each HomFam family where the forward and reverse TC core scores are different. 68 HomFam protein families with \( n = 100 \) samples for each family.

Effect is also surprising and mainly shows up when the numbers of sequences grows large. It can, nonetheless, be seen in data sets of the order of a hundred sequences or so.

On examining the distance matrices generated by some widely used multiple sequence alignment packages, it can be seen that these become increasingly dominated by tied values. The more sequences being aligned, the greater the percentage of the scores in a distance matrix that are duplicates of other scores. This effect can be traced to the use of \( k \)-tuple scores for computing these distances. For sequences of a given length, there is a finite and relatively small number of possible scores that can be generated. For shorter length sequences, the number of possible distances is also reduced. If real alignment scores using an amino acid weight matrix such as BLOSUM are used, the number of possible scores is still finite although much greater than with \( k \)-tuple distances. Given enough sequences though, there will inevitably be many tied values in a distance matrix. The use of such alignment scores is limited however, to relatively small datasets as they are expensive to compute, as was seen with Mafft G-INS-1 in Figure 1. For really big alignments, of many thousands
of sequences, there is currently little alternative to the use of $k$-tuple or word based scores at some stage of the progressive alignment procedure. Iteration, as carried out by Clustal Omega, Mafft and Muscle can help as the later alignments can use real alignment scores but these are very expensive computationally and do not eliminate tied scores. It is also possible to mitigate the alignment instability by, say, ordering the input sequences lexicographically before calculating the $k$-tuple scores. However, while this will result in a consistent alignment being produced, it is difficult to justify from a biological point of view why the alignment produced by such an ordering should be chosen out of the numerous alternatives.

The solution to this issue is not clear-cut, given the need for speed in the construction of guide trees outlined in this chapter's Introduction. Sievers et al. (2013) have shown previously that the accuracy of progressive alignment decreases markedly with very large datasets, and this was assumed to be due to the greedy nature of the progressive alignment algorithm. The results here show that progressive alignment also produces alignments that have a strong dependence on the sequence order in the input file. The use of chained guide trees described in Chapter 2 can help improve accuracy but will still have a strong dependence on input file sequence order.

One final point should be noted: In Chapter 2, random chained guide trees were shown to generate alignments of equal quality to those where the ordering of the sequences in the chained guide tree was optimised based on the distance measure between each pair of sequences. However, as was shown in Figure 3.2, the vast majority of these pairwise distance measures were duplicates, particularly when calculated on larger numbers of sequences. Hence, even though the optimisation routines reached stability, there were a myriad possible sequence orderings that could have met the same optimisation criteria. Given this, the similarity between the random and optimised orderings of sequences is not as surprising as it first appeared to be.

This point will be examined in more detail in the next chapter.
Chapter 4

Dynamic Ordering of Sequences

4.1 Introduction

In Chapter 2 it was shown that, for larger alignments, chained guide trees produced high quality alignments when measured against structural benchmarks. It was also shown that optimising the ordering of sequences on the leaves of the guide tree using the $k$-tuple distance measure gave little improvement when compared to randomly ordered sequences. Subsequently, Chapter 3 showed that the $k$-tuple distance is a coarse-grained measure of sequence similarity, and once the number of sequences to be aligned goes beyond a relatively small number the vast majority of pairwise distance measures will have tied values. Hence, while the optimisation routines in Chapter 2 did reach a local minimum (or maximum), there were so many different orderings with the same overall distance measures that the sequence order chosen was essentially a random ordering.

This chapter attempts to resolve two questions from the preceding chapters. While a number of different sequence ordering strategies were compared to random ordering in Chapter 2, all of them used a distance measure based on $k$-tuple scores. In addition, in Chapter 3 the instability in the alignment algorithms was shown to be due to tied distance measures, again based on the $k$-tuple scores. In this chapter, an alternative to the $k$-tuple similarity score is proposed. Using this new similarity measure, sequences can be ordered in a chained guide tree that results in higher quality alignment when measured using a structural benchmark. In addition, this similarity measure does not suffer from the same number of tied values in tests conducted using hundreds or small numbers of thousands of sequences, and so the impact of the sequence ordering on the alignments produced is removed.

This analysis uses a HMM bitscore to indicate how similar individual or groups of sequences
are to each other. The generation of HMMs and comparison of sequences uses the HMMER profile hidden Markov model package (HMMER). The HMM bitscore calculated by HMMER is the $\log_2$ of the probability of the sequences matching divided by the probability according to a null model, where the null model consists of residues drawn from a background frequency distribution hardcoded as the mean residue frequencies in Swiss-Prot 50.8 [HMMER Users Guide, p39]. HMMs have been widely used for sequence comparison and protein database searching, and by using position-specific information can detect subtle similarities between sequences (Krogh et al., 1994; Eddy, 1996). Hence, HMM comparison should give a much finer-grained measure of similarity (or distance) between sequences than the $k$-tuple score.

In addition, HMMER will also convert an existing alignment to a profile HMM. This allows unaligned sequences to be re-compared to the growing alignment rather than relying on the distance between pairs of sequences calculated at the start of the alignment process. In this way, the most similar sequences can be aligned to the partial alignment at each step in the progressive alignment, which is the intention behind the use of a guide tree.

The idea of a dynamically updated guide tree was originally proposed by Wheeler and Kececioglu (2007): “...distances are derived from sequences only at initialization. When group $ab$ is formed, the new distances $d_{abc}$ are calculated from original sequence distances, which ignores the constraints on sequence pairs across groups $ab$ and $c$ imposed by the alignments for these groups [i561]”. Wheeler and Kececioglu evaluated a number of methods that take these constraints into consideration. The best of these methods they called DAD for dynamic alignment distance, but unfortunately this did not perform as well on their tests as Neighbour-Joining (Saitou and Nei, 1987), UPGMA (Sokal and Michener, 1958) or Minimum Spanning Tree. In their view, this was due to alignments against larger groups being more constrained than those against smaller groups, leading to larger groups tending to have higher distances, which in turn caused smaller groups to be merged first.

The approach proposed here differs from Wheeler and Kececioglu’s. This method uses an implicit chained guide tree topology. The initial pair of sequences to be aligned are selected based on the highest bitscore between each pair of sequences. The resulting alignment is converted into a profile HMM using HMMER, the remaining sequences are compared with it, and the sequence with the highest bitscore is the next selected to be aligned against the partial alignment. This process is then repeated until the full alignment has been generated. By constantly selecting the next most
similar sequence, an overall guide tree is not created in advance. Instead, the order in which the sequences are aligned emerges during the alignment process itself.

This approach, termed dynamic ordering here aims to include the benefits shown previously of chained guide trees, without the issues associated with calculating the distance between sequences using the \( k \)-tuple score. Dynamic ordering uses a HMM bitscore instead of the \( k \)-tuple score between sequences to determine their degree of similarity, and the range of possible values for the bitscore should be far higher than the quite-limited range for \( k \)-tuple scores. In addition, the bitscores in dynamic ordering enforce an ordering on the sequences, and sequences with tied bitscores will be added to the growing alignment in close succession, assuming that the alignment does not change too much when the first sequence is added. Tied \( k \)-tuple scores in the default methods, on the other hand, will cause sequences to be added to one branch of the guide tree or the other, resulting in the particular sequence being aligned with very different sequences. Hence, dynamic ordering should reduce if not remove the dependency that alignments created with the existing methods have on the ordering of the sequences in the input file.

One of the advantages, however, of using \( k \)-tuple scores is speed. The method proposed here will be considerably slower, at least in the basic implementation described below. The initial all-against-all comparison of sequences to find the highest scoring pair has a time complexity of \( O(LN) \) to create the profiles for \( N \) sequences of length \( L \), followed by the actual comparisons with complexity of \( O(LN^2) \). Determining the most similar sequences takes an additional \( O((LN + L'N)N) \), where \( LN \) is again required to create the HMM profile at each step and \( L'N \) time is required to score each of the remaining \( N \) sequences against a profile of length \( L' \). Both of these are repeated to determine each next-most-similar sequence. In addition, the analysis presented relies heavily on bash scripting and file I/O, rather than data manipulation within a single program. While the time needed is an important requirement, both the quality and stability of the alignments generated will have a higher priority in this analysis.

### 4.2 Methods

#### 4.2.1 HomFam

The analysis and results presented in this chapter use the HomFam (Blackshields et al., 2010) structural alignment benchmark system to measure the quality of the alignments produced by the various
alignment programs. As before, HomFam combines single-domain Pfam (Sonnhammer et al., 1997) families which have at least 5 members with known structures in a HOMSTRAD (Mizuguchi et al., 1998b) structural alignment. A more recent Pfam release, version 28, and an updated version of HOMSTRAD are used here than in the results presented in previous chapters, and the alignment quality is measured using the TC score of the BAliSCORE program (Thompson et al., 1999) based on the core columns of the reference sequences only.

One of the aims of this chapter is to devise a way to eliminate the alignment algorithm instability that was described in Chapter 3. In that analysis, all duplicate sequences were removed from each protein family, as identical sequences would impact the minimum values in the distance matrix and the guide tree constructed from the matrix. For the same reason, duplicate sequences are also removed here.

This analysis uses a HMM bitscore to indicate how similar individual or groups of sequences are to each other. As longer similar sequences will tend to have a higher bitscore than shorter similar sequences just due to their sequence length, it is necessary to ensure that both the Pfam sequences and the HOMSTRAD reference sequences have similar lengths. If a wide range of sequence and reference lengths are used, there may be a tendency for sequences to be stratified into longer and shorter sequence groups, which could have an impact on the reported TC scores if the reference sequences are confined to a narrow band within the full range of sequence lengths.

Figure 4.1 shows that there are a number of protein families where the HOMSTRAD reference sequence lengths are either outside the range of sequence lengths for the corresponding Pfam family, or at one extreme of the length range. For this reason, both Pfam and HOMSTRAD sequences outside the common range of sequence lengths for that protein family were removed. Protein families that then had less than five HOMSTRAD reference sequences were completely removed, leaving a total of 82 families, details of which are given in Table 4.1.

### 4.2.2 Algorithm

The approach proposed in this chapter is based on the principle of aligning the most similar sequences together, with the determination of what constitutes the “most similar” being re-evaluated at each step in the alignment process. The algorithm outline is as follows:

1. The most similar sequences are first selected based on the similarity measured between each pair of sequences.
Figure 4.1: Length of the Pfam data and HOMSTRAD reference sequences for each HomFam protein family before deduplication and filtering by sequence length. Top: Pfam version 25 and HOMSTRAD as of 2011-06-13. Bottom: Pfam version 28 and HOMSTRAD as of 2015-04-23.
<table>
<thead>
<tr>
<th>Protein Family</th>
<th>Initial Seqs</th>
<th>Final Seqs</th>
<th>No of Refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ace</td>
<td>25,777</td>
<td>2,144</td>
<td>6</td>
</tr>
<tr>
<td>aldosed</td>
<td>17,219</td>
<td>8,896</td>
<td>7</td>
</tr>
<tr>
<td>alpha-amylase_C</td>
<td>16,801</td>
<td>1,877</td>
<td>21</td>
</tr>
<tr>
<td>asp</td>
<td>8,888</td>
<td>2,397</td>
<td>13</td>
</tr>
<tr>
<td>az</td>
<td>4,260</td>
<td>1,683</td>
<td>28</td>
</tr>
<tr>
<td>biotin_lipoyl</td>
<td>114,752</td>
<td>43</td>
<td>6</td>
</tr>
<tr>
<td>blmb</td>
<td>141,386</td>
<td>6,829</td>
<td>6</td>
</tr>
<tr>
<td>bv</td>
<td>166</td>
<td>75</td>
<td>5</td>
</tr>
<tr>
<td>cah</td>
<td>6,251</td>
<td>2,399</td>
<td>5</td>
</tr>
<tr>
<td>CBM_20</td>
<td>2,560</td>
<td>230</td>
<td>8</td>
</tr>
<tr>
<td>ChlBD</td>
<td>2,168</td>
<td>161</td>
<td>5</td>
</tr>
<tr>
<td>cyclo</td>
<td>41,137</td>
<td>4,174</td>
<td>6</td>
</tr>
<tr>
<td>cys</td>
<td>12,552</td>
<td>6,111</td>
<td>13</td>
</tr>
<tr>
<td>cyt3</td>
<td>640</td>
<td>114</td>
<td>6</td>
</tr>
<tr>
<td>cytB</td>
<td>12,139</td>
<td>697</td>
<td>6</td>
</tr>
<tr>
<td>cytC</td>
<td>28,796</td>
<td>2,127</td>
<td>11</td>
</tr>
<tr>
<td>DMRL_synthase</td>
<td>18,312</td>
<td>69</td>
<td>5</td>
</tr>
<tr>
<td>egf</td>
<td>35,435</td>
<td>912</td>
<td>9</td>
</tr>
<tr>
<td>fabp</td>
<td>3,152</td>
<td>1,128</td>
<td>17</td>
</tr>
<tr>
<td>fer2</td>
<td>69,259</td>
<td>225</td>
<td>13</td>
</tr>
<tr>
<td>flav</td>
<td>40,676</td>
<td>3,372</td>
<td>6</td>
</tr>
<tr>
<td>fn3</td>
<td>1,040</td>
<td>592</td>
<td>14</td>
</tr>
<tr>
<td>gfh10</td>
<td>4,027</td>
<td>1,414</td>
<td>5</td>
</tr>
<tr>
<td>gfh11</td>
<td>1,334</td>
<td>567</td>
<td>5</td>
</tr>
<tr>
<td>gfh18</td>
<td>18,714</td>
<td>1,373</td>
<td>5</td>
</tr>
<tr>
<td>gfh1</td>
<td>44,198</td>
<td>11,898</td>
<td>8</td>
</tr>
<tr>
<td>gfh22</td>
<td>1,369</td>
<td>746</td>
<td>12</td>
</tr>
<tr>
<td>gfh5</td>
<td>11,685</td>
<td>3,562</td>
<td>6</td>
</tr>
<tr>
<td>gtp</td>
<td>29,109</td>
<td>8,435</td>
<td>8</td>
</tr>
<tr>
<td>Haloperoxidase</td>
<td>15,309</td>
<td>1,826</td>
<td>9</td>
</tr>
<tr>
<td>HGTP_anticodon</td>
<td>71,621</td>
<td>4,340</td>
<td>6</td>
</tr>
<tr>
<td>hla</td>
<td>33,156</td>
<td>5,690</td>
<td>5</td>
</tr>
<tr>
<td>HLH</td>
<td>22,320</td>
<td>704</td>
<td>5</td>
</tr>
<tr>
<td>HMG_box</td>
<td>13,187</td>
<td>549</td>
<td>5</td>
</tr>
<tr>
<td>hom</td>
<td>35,751</td>
<td>12,123</td>
<td>8</td>
</tr>
<tr>
<td>hormone_rec</td>
<td>9,809</td>
<td>27</td>
<td>5</td>
</tr>
<tr>
<td>hpr</td>
<td>35,448</td>
<td>542</td>
<td>5</td>
</tr>
<tr>
<td>hr</td>
<td>9,896</td>
<td>3,365</td>
<td>5</td>
</tr>
<tr>
<td>icd</td>
<td>44,384</td>
<td>8,555</td>
<td>5</td>
</tr>
<tr>
<td>il8</td>
<td>2,717</td>
<td>67</td>
<td>8</td>
</tr>
<tr>
<td>ins</td>
<td>1,725</td>
<td>124</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Protein Family</th>
<th>Initial Seqs</th>
<th>Final Seqs</th>
<th>No of Refs</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>20,053</td>
<td>1,525</td>
<td>5</td>
</tr>
<tr>
<td>kazal</td>
<td>2,638</td>
<td>425</td>
<td>6</td>
</tr>
<tr>
<td>kringle</td>
<td>3,626</td>
<td>1,281</td>
<td>9</td>
</tr>
<tr>
<td>kunitz</td>
<td>8,677</td>
<td>1,141</td>
<td>10</td>
</tr>
<tr>
<td>LIM</td>
<td>24,277</td>
<td>3,305</td>
<td>5</td>
</tr>
<tr>
<td>lipocalin</td>
<td>3,152</td>
<td>261</td>
<td>14</td>
</tr>
<tr>
<td>MHC_II_N</td>
<td>3,900</td>
<td>2,263</td>
<td>11</td>
</tr>
<tr>
<td>mmp</td>
<td>6,707</td>
<td>1,159</td>
<td>6</td>
</tr>
<tr>
<td>msb</td>
<td>17,045</td>
<td>5,288</td>
<td>8</td>
</tr>
<tr>
<td>myb_DNA-binding</td>
<td>28,131</td>
<td>2,683</td>
<td>5</td>
</tr>
<tr>
<td>p450</td>
<td>30,036</td>
<td>8,891</td>
<td>12</td>
</tr>
<tr>
<td>PDZ</td>
<td>48,255</td>
<td>1,963</td>
<td>6</td>
</tr>
<tr>
<td>phospholipin</td>
<td>2,016</td>
<td>98</td>
<td>18</td>
</tr>
<tr>
<td>profilin</td>
<td>1,710</td>
<td>666</td>
<td>5</td>
</tr>
<tr>
<td>response_reg</td>
<td>74,211</td>
<td>67,633</td>
<td>13</td>
</tr>
<tr>
<td>RdFadensenate</td>
<td>131,926</td>
<td>344</td>
<td>6</td>
</tr>
<tr>
<td>rnasemam</td>
<td>1,244</td>
<td>889</td>
<td>6</td>
</tr>
<tr>
<td>rrm</td>
<td>97,323</td>
<td>2,433</td>
<td>14</td>
</tr>
<tr>
<td>RubisCO_large</td>
<td>81,136</td>
<td>16,138</td>
<td>6</td>
</tr>
<tr>
<td>rub</td>
<td>12,930</td>
<td>2,398</td>
<td>5</td>
</tr>
<tr>
<td>rvp</td>
<td>168,023</td>
<td>415</td>
<td>6</td>
</tr>
<tr>
<td>scorpotoxin</td>
<td>548</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>serbact</td>
<td>45,666</td>
<td>1,563</td>
<td>5</td>
</tr>
<tr>
<td>sermam</td>
<td>45,666</td>
<td>15,421</td>
<td>27</td>
</tr>
<tr>
<td>serpin</td>
<td>8,202</td>
<td>2,839</td>
<td>8</td>
</tr>
<tr>
<td>sh2</td>
<td>13,161</td>
<td>51</td>
<td>11</td>
</tr>
<tr>
<td>sh3</td>
<td>18,736</td>
<td>1,263</td>
<td>15</td>
</tr>
<tr>
<td>slectin</td>
<td>3,115</td>
<td>1,278</td>
<td>5</td>
</tr>
<tr>
<td>sodcu</td>
<td>12,155</td>
<td>104</td>
<td>7</td>
</tr>
<tr>
<td>sti</td>
<td>1,122</td>
<td>579</td>
<td>5</td>
</tr>
<tr>
<td>subt</td>
<td>9,313</td>
<td>2,772</td>
<td>11</td>
</tr>
<tr>
<td>Sulfortransfer</td>
<td>8,739</td>
<td>4,308</td>
<td>5</td>
</tr>
<tr>
<td>tgb</td>
<td>5,158</td>
<td>2,721</td>
<td>8</td>
</tr>
<tr>
<td>thioreductase</td>
<td>88,141</td>
<td>24,193</td>
<td>6</td>
</tr>
<tr>
<td>TiG</td>
<td>12,418</td>
<td>4,648</td>
<td>6</td>
</tr>
<tr>
<td>tim</td>
<td>24,559</td>
<td>1,127</td>
<td>10</td>
</tr>
<tr>
<td>tms</td>
<td>19,061</td>
<td>2,704</td>
<td>5</td>
</tr>
<tr>
<td>toxin</td>
<td>794</td>
<td>603</td>
<td>20</td>
</tr>
<tr>
<td>TyrKc</td>
<td>30,723</td>
<td>17,676</td>
<td>7</td>
</tr>
<tr>
<td>uce</td>
<td>17,227</td>
<td>9,050</td>
<td>13</td>
</tr>
<tr>
<td>zf-CCHH</td>
<td>99,707</td>
<td>1,164</td>
<td>11</td>
</tr>
</tbody>
</table>

Table 4.1: Initial number of sequences, final number of sequences after deduplication and filtering by sequence length and number of reference sequences in each HomFam protein family.
(a) An HMM profile is created for each individual sequence using \texttt{HMMBUILD}.

(b) All profiles are concatenated and prepared for scoring with \texttt{HMMPRESS}.

(c) The full set of unaligned sequences is scored against this collection of HMM profiles using \texttt{HMMSCAN}, and the sequence-profile combination with the highest bitscore (most similar) is selected, obviously excluding sequences scored against their own profile.

2. The two most similar sequences are aligned using a standard alignment with the given alignment program.

3. The remaining unaligned sequences are scored against the current alignment, and the highest-scoring sequence is then added to the alignment using a profile alignment.

(a) The current alignment is converted to a HMM profile using \texttt{HMMBUILD}, and the profile is prepared for scoring with \texttt{HMMPRESS}.

(b) The remaining unaligned sequences are scored against this profile using \texttt{HMMSCAN} and the most similar sequence, based on its HMM bitscore, is extracted.

(c) This sequence is added to the growing alignment using a profile alignment to preserve the alignment that has been generated up to this point.

(d) In a small number of cases, using the default threshold values \texttt{HMMSCAN} fails to find any similar sequences. In this situation, the first sequence in the remaining sequences file is selected and aligned.

(e) This process is repeated until all sequences have been aligned.

4.2.3 Alignment programs

The alignment programs Clustal Omega (Sievers et al., 2011), Mafft (Katoh et al., 2002) and Muscle (Edgar, 2004b) are again used in this chapter. These were previously used in Chapters 2 and 3, and were selected based on their widespread use, their ability to use an externally-defined guide tree and to align more than a thousand protein sequences. Unfortunately, Kalign2 (Lassmann et al., 2009) could not be used even though its alignments depend on the order of the input sequences, as it cannot accept an external guide tree.

The default parameters were used where possible for all alignment programs, except when it was necessary to specify an input guide tree or to add a sequence to an existing (aligned) profile. In Mafft,
the FFT-NS-2 algorithm was used in all cases, and the `newick2mafft.rb` ruby script, downloaded from the Mafft website, was used to convert all externally-generated guide trees into Mafft format. In addition, the ruby script `makemergetable.rb` also downloaded from the Mafft website, was used to define which sub-alignments to merge into a single multiple sequence alignment. The `--unweight` option was also used to ensure branch lengths on external guide trees were ignored.

It should be noted that this is a later version of Mafft than used in the previous chapters. In a recent article, Yamada et al. (2016) announced that since Mafft version 7.1, a reduced gap cost is used resulting in alignments with more and longer gaps. The article’s topic was to examine the effect of chained guide trees, and when the tests were repeated with the new version of Mafft, they found that random chained guide trees performed poorer than the default trees when measured on the HomFam benchmark. Curiously the new version of Mafft using random chained guide trees performed better than with the default trees when measured using ContTest (Fox et al., 2016).

With Muscle, the number of iterations was limited to 2 rather than the default of 16. This is the number of iterations recommended by the program’s authors for large datasets. A bug was found in Clustal Omega when doing a profile alignment. A check at line 1115 of the program `mymain.c` was failing as it mistakenly thought the input profile was not aligned. This check was commented out to allow the program to run. For comparison purposes, each alignment program was also run with chained guide trees, and these trees were created using a separate utility program.

Finally, the defaults for both Mafft and Muscle use 2 iterations. From each program’s documentation, in the second iteration a guide tree is created using the results from the first alignment iteration, and this new guide tree defines the alignment order. It is not clear what happens in either program when iterating an alignment of a sequence against a profile. The intention is that the profile is already aligned and so should remain fixed. However, it is possible that after the new sequence has been aligned against the profile, the combined alignment is used to create a new guide tree containing all the individual sequences assigned to \( N \) leaves where \( N \) is the total number of sequences, rather than just 2 leaves for the profile and the new sequence being added in this alignment step.

The program versions and runtime parameters are as follows:

**Clustal Omega** (v1.2.0-r289):

- **default**: `--i <sequence file> --o <alignment file>`
- **guidetree**: `--i <sequence file> --o <alignment file> --guidetree-in=<guidetree file>`
- **profile**: `--profile1 <profile1 file> --profile2 <profile2 file> --o <alignment file>`
Mafft FFT-NS-2 (v7.245):

default: --anysymbol <sequence file> > <alignment file>

guidetree: --anysymbol --treein <guidetree file> --unweight <sequence file> > <alignment file>

profile: --merge <merge table> --anysymbol <sequence file> > <alignment file>

Muscle (v3.8.31):

default: -in <sequence file> -out <alignment file> -maxiters 2

guidetree: -in <sequence file> -out <alignment file> -usetree_nowarn <guidetree file> -maxiter 2

profile: -profile -in1 <sequence file 1> -in2 <sequence file 2> -out <alignment file> -maxiters 2

HMMER (v3.1b2):

hmmbuild: <hmm profile file> <alignment filename>

hmmpress: <hmm profile file>

hmmscan: --tblout <output file> <hmm profile file> <sequences file>

4.2.4 Supporting material

Details of utility programs and scripts used are given in Appendix A.

4.3 Results

4.3.1 Static versus dynamic ordering

As a first step, it is necessary to determine how much of an impact, if any, the recalculation of the similarity scores during the alignment process has on the order in which the sequences are aligned. It is not feasible to compare the alignment ordering of this new approach with the order inherent in default guide trees, and chained trees so far have either been randomly ordered or ordered based on k-tuple scores.

For the purposes of comparison with dynamic ordering, a static ordering of sequences based on their HMM bitscores was created. 100 sequences were selected at random from the HomFam families ace (alpha beta-hydrolase, Pfam accession number PF00135), p450 (Cytochrome P450, Pfam accession number PF00067) and subt (subtilase, Pfam accession number PF00082). The
two most similar sequences in each family were found based on their highest HMMER bitscore as described above. The remaining sequences were scanned, and the one most similar to any of the already selected sequences (based on the HMMER bitscore) was selected next. This step was repeated until all sequences were selected, and the order in which the sequences were selected defined the static ordering of the sequences.

The dynamic ordering of the sequences was determined as described in Section 4.2.2 above, and for each sequence dynamically ordered, its position in the static ordering was also determined. Figure 4.2 (top) shows the degree of similarity between these two orderings for each of the three alignment programs. While the orders started off quite similar, there was a reasonable amount of variation between them, indicating that the concept of aligning most similar sequences changed over the course of the alignment process. The similarity between the two orders towards the end of the alignment process indicated that there were sequences which were quite different from the starting sequences and from the alignment being created. There was also some similarity in the ordering pattern across the three alignment programs indicating that, at least in general terms, the three programs produced similar alignments.

The process was repeated with 500 sequences for the same three protein families, and the results are shown in the bottom panel of Figure 4.2. Again the two different orderings started off quite similar, except for alpha beta-hydrolase and Clustal Omega. However, the orderings deviated considerably more than with the smaller number of sequences, which is to be expected as the alignment will change more as more sequences are added to it. The plots for all three alignment programs again show a broadly similar pattern for each protein family, but not to the same extent as with 100 sequences.

Overall, however, the results are quite clear. Aside from a small number of sequences at the start of the alignments, there is a divergence between the static and dynamic ordering of the sequences, and this divergence is more pronounced with larger datasets. Hence, if the approach is to align the most similar sequences first, the determination of what constitutes “most similar” should be updated over the course of the alignment process.

4.3.2 Balibase

The BAliBASE (Thompson et al., 2005) structural benchmark dataset consists of 217 reference alignments across 5 Reference sets, with Reference Set 1 divided into 2 subsets. For each reference
Figure 4.2: Static versus dynamic ordering for (top) 100 sequences and (bottom) 500 sequences randomly selected from ace (alpha beta-hydrolase, Pfam accession number PF00135), p450 (Cytochrome P450, Pfam accession number PF00067) and subt (subtilase, Pfam accession number PF00082) using Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations). The blue diagonal line shows where the two orders are the same.
alignment dataset, the sequences were aligned using the default versions of Clustal Omega, Mafft FFT-NS-2 and Muscle (2 iterations). The sequences were also aligned using the dynamic ordering approach described above. As a third option, alignments were also created using random chained guide trees, where the sequences in each reference alignment were randomly shuffled before a chained guide tree was constructed using a small utility program. As there were a reasonably large number of reference alignments and many of these had quite a small number of sequences, each set of sequences was only sampled once. The quality of all alignments were measured using the *bali_score* program on just the core columns, and the TC scores were compared using the non-parametric one-tailed paired Wilcoxon signed-rank test. The results are shown in Figure 4.3.

![Figure 4.3: TC scores for the different BAliBASE reference sets using default guide trees (Def), dynamic ordering (Dyn) and random chained guide trees (Rnd) for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations). (*P < 0.05, **P < 0.01, ***P < 0.001, Red: Default TC scores greater than Dynamic).](image)

In the majority of scenarios, alignments using dynamic ordering had significantly higher quality score than that for alignments using random chained guide trees, particularly for reference sets BB12 (references with at least four equidistant sequences, with any two sequences sharing 20–40% identity) and BB40 (references with large N/C-terminal extensions). However, as was shown in Chapter 2, the quality improvements from using random chained guide trees only come into effect with large alignments of approximately 1,000 sequences or more. The comparison of dynamic
ordering with the default alignments is somewhat more mixed. Again, dynamic ordering does quite well with reference set BB12, but no difference is detected with reference sets BB20 (families with one or more highly divergent “orphan” sequences), BB40 and BB50 (sequences with large internal insertions). Interestingly, the default alignments of all three alignment programs give higher quality results with reference set BB30 (families containing divergent subfamilies), perhaps because the guide trees in the default alignments tend to cluster the subfamilies together, thereby ensuring that individual sequences from each subfamily are first aligned with similar sequences.

4.3.3 HomFam

The comparison between alignments produced using the default versions of the alignment programs, using random chained guide trees, and using dynamic ordering alignments was repeated with the HomFam structural benchmark, after the sequences had been deduplicated and filtered by length. For each HomFam family, random samples of various size were taken, and the sample was aligned using Clustal Omega, Mafft (FFT-NS-2) and Muscle (2 iterations). 100 random samples were drawn for each sample size, and families with an insufficient number of Pfam and HOMSTRAD sequences combined were excluded for that particular sample size. Each alignment program was run three times with each sample: using (i) the default parameter values; (ii) a randomly-ordered chained guide tree; and (iii) the dynamic ordering method. The quality of each alignment was determined using the TC score measured on the reference sequence core columns only, and the TC scores were compared using the non-parametric one-tailed paired Wilcoxon signed-rank test.

The results from the three different alignment options were examined as three pairwise comparisons. As an initial test, default settings and random chained guide trees were compared to see if the quality improvement shown in Chapter 2 for random chained guide trees still applied with the modified dataset used in this chapter, and the updated version of Mafft described earlier.

There was little overlap in the data used to produce the results shown in Figure 4.4 and the previous results shown in Figure 2.10, because of the deduplication and filtering of the sequences. However, the general trend of alignments using random chained guide trees having higher quality with larger numbers of sequences was approximately the same. Random chained guide trees performed better here for Clustal Omega than with the previous dataset, and the default trees performed worse — 46% and 25% of families here respectively for 1,000 sequences versus 24% and 51% previously for 1,024 sequences. With Muscle, the number of families where random chained trees gave
Figure 4.4: Comparison of TC scores obtained for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with default settings and randomly chained guide trees for different dataset sizes across all HomFam families. Each point is the mean TC score for that dataset. The colour coding is determined using a one-tailed paired Wilcoxon signed-rank test. The diagonal is for reference to indicate where the two TC scores are equal. (Significance level $\alpha = 0.01$, 100 samples per family and number of sequences.)
higher quality scores was approximately the same, 46% here for 1,000 sequences versus 44% previously for 1,024 sequences. Muscle previously showed a considerable increase in the number of families where random chained trees gave higher quality scores when the sample sizes increased to 2,048 and 4,096 sequences, but there were insufficient data here for comparison purposes. Interestingly, the Mafft results here were also quite similar to those reported previously. With 1,000 sequences, for 44% of families random chained guide trees gave higher TC scores here and default trees performed better for 40% of families. In comparison, with 1,024 sequences previously these figures were 41% and 39%. However, Yamada et al. (2016) found that with the version of Mafft used here default alignments had higher quality scores than those produced using random chained guide trees. It is unclear if this difference in findings was due to the filtering of the data used to produce these results. It may also be due to a difference in the presentation of the results, with Yamada et al. reporting the average TC scores across multiple families, whereas the percentages given here are based on the count of families with a significant difference in TC scores.

The results of dynamic ordering and random chained guide trees are shown in Figure 4.5. As dynamic ordering, in essence, also uses a chained guide tree topology, the purpose of this test was to determine what effect ordering has on the alignment quality across different sample sizes.

For the smallest sample sizes, dynamic ordering gave higher quality scores for the vast majority of protein families. As the number of sequences increased, the results varied somewhat across the different alignment programs. With Mafft, the number of families where dynamic ordering outperformed random chained guide trees quickly dropped off as the number of sequences increased. The number of families where random chained trees scored best increased with the number of sequences, as did the number of neutral families where neither alignment approach gave a significantly better alignment quality. With Muscle, the number of neutral families also increased as the sample size grew larger. Also, the number of families where dynamic ordering performed better remained quite large, at least up until 1,000 sequences. However, with Clustal Omega, dynamic ordering performed considerably better than random chained guide trees across all sample sizes. The number of neutral families remained at about 20% regardless of sample size.

The main focus of this chapter, however, is whether dynamic ordering performs better than each alignment program’s default settings. The results given in Figure 4.6 are somewhat mixed. For all alignment programs, dynamic ordering performed worse than the default method for the smaller two sample sizes, and the alignment quality increased as the number of sequences also increased.
Figure 4.5: Comparison of TC scores obtained for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with dynamic ordering and randomly chained guide trees for different dataset sizes across all HomFam families. Each point is the mean TC score for that dataset. The colour coding is determined using a one-tailed paired Wilcoxon signed-rank test. The diagonal is for reference to indicate where the two TC scores are equal. (Significance level $\alpha = 0.01$, 100 samples per family and number of sequences.)
Corresponding to this trend, there was a general falling off in the number of protein families where the default alignments gave higher quality alignments. There was some variation across the alignment programs, however. For Mafft, the default alignment outperformed dynamic ordering. The results were quite similar to those obtained with Mafft when comparing the default option and random chained guide trees, which is to be expected given the number of neutral families when comparing dynamic ordering and random chained guide trees. With Clustal Omega on the other hand, for datasets of 250 or more sequences, dynamic ordering produced far more families with higher TC scores than the default alignment option. This is in contrast to the results reported in Chapter 2 where the default alignments gave higher quality alignments for the majority of protein families with samples of 1,024 sequences. Part of this difference may due to the data deduplication and filtering used in this chapter, as random chained guide trees performed better here that default alignments. In addition, as was shown above, dynamic ordering performed better in considerably more families than random chained guide trees.

One final point to note is that in almost all situations the number of families where the difference in TC scores was not statistically significant is quite low, even with a significance level, $\alpha$, value of 0.01. Increasing $\alpha$ to 0.05 gave only a marginal change to the number of families in each category.

### 4.3.4 Instability

The second objective of this chapter was to eliminate the instability in progressive alignment algorithms, due to the construction of the guide tree based on insufficiently granular $k$-tuple scores. The rationale for using HMM bitscores and the expected benefits from this were outlined in the chapter’s introduction. To determine whether these benefits can be realised, the initial test described in Chapter 3 was repeated here using the latest deduplicated and filtered HomFam data.

For each HomFam family, a random selection of sequences was made and combined with the family’s HOMSTRAD reference sequences to give 1,000 sequences. Families with an insufficient number of sequences were excluded, leaving a total of 52 families. The 1,000 sequences were randomly shuffled, and a default alignment was made with Clustal Omega, Mafft (FFT-NS-2) and Muscle (2 iterations). The order of the sequences in the input file was reversed and the alignments repeated with the same parameters. The difference between the quality scores of the two alignments for each alignment program was calculated. This process was then repeated 10 times for each of the 52 HomFam families, and the results are presented in the left-hand column of Figure 4.7.
Figure 4.6: Comparison of TC scores obtained for Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations) with default settings and dynamic ordering for different dataset sizes across all HomFam families. Each point is the mean TC score for that dataset. The colour coding is determined using a one-tailed paired Wilcoxon signed-rank test. The diagonal is for reference to indicate where the two TC scores are equal. (Significance level $\alpha = 0.01$, 100 samples per family and number of sequences.)
Dynamic ordering alignments were made with each of the three alignment programs for all the same datasets, and the difference between the quality scores of the forward and reversed sequence files calculated. These results are shown in the right-hand column of Figure 4.7.

The results for the default alignment options show a considerable degree of variance, as was shown in Figure 3.1 previously. With dynamic ordering, however, the vast majority of protein families showed no variance in their TC scores for any of the samples. Where a difference in TC scores did occur, this was due to the default values of the HMMER thresholds that were used. In a very small number of cases, HMMER did not find a sequence that was sufficiently similar to HMM profile in question. In those situations, the \texttt{bash} script controlling the processing flagged that no match was found, but as a sequence had to be added, it picked the first sequence of those still to be aligned. Changing the order of the input file then resulted in a different sequence being selected, which caused different subsequent sequences to be selected. To demonstrate this, the HomFam family fabp (Fatty acid binding protein-like, Pfam accession number PF00061) is the protein family towards the left of the HomFam families where all samples for Clustal Omega dynamic ordering in Figure 4.7 have different forward and reverse TC scores. All samples produced warning messages for dynamic ordering, and the first five when the forward samples were processed with Clustal Omega were:

- \texttt{fabp 129 No matches with profile, picked >G3NQ01_GASAC__6 -134}
- \texttt{fabp 127 No matches with profile, picked >Q4VBT8_DANRE__6 -134}
- \texttt{fabp 131 No matches with profile, picked >A0A034WXG8_RHIMP__6 -131}
- \texttt{fabp 128 No matches with profile, picked >H0X2B1_OTOGA__6 -134}
- \texttt{fabp 127 No matches with profile, picked >G5BF59_HETGA__6 -132}

And when the sequences were reversed, the warning messages that were issued from the same samples showed that different sequences were selected:

- \texttt{fabp 129 No matches with profile, picked >D57666_PARCR__6 -133}
- \texttt{fabp 127 No matches with profile, picked >D0EHJ2_PHACC__6 -132}
- \texttt{fabp 133 No matches with profile, picked >F1NDE7_CHICK__6 -132}
- \texttt{fabp 128 No matches with profile, picked >B5B7Q8_SCHJA__5 -132}
- \texttt{fabp 127 No matches with profile, picked >K9IGW2_DESRO__5 -138}

The number after the protein family name is the alignment step where a match could not be found between the profile and the unaligned sequences. Also, the ‘/’ character in sequence names has been replaced by ‘\_\_’ as the \texttt{bash} scripts create temporary files named with the sequence name. fabp has only 1,128 in total after filtering and deduplications, so all samples of 1,000 sequences will contain more or less the same sequences. Hence, the missing sequences occurred at more or less the same alignment step in each of the samples. This source of instability can be avoided by...
Figure 4.7: The difference in the TC core scores for 1,000 randomly-selected sequences and in reverse order aligned with Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations), using default options (left) and dynamic ordering (right). (52 HomFam protein families, 10 samples per family).
lowering or removing the HMMER similarity threshold, so that one sequence has to be deemed the sequence next most similar to the alignment.

Interestingly, for the third sample listed, the position in which no matches were found is larger for the sample being processed in the reverse direction than forward. For extra sequences to be included, there must have been tied HMM bitscores, and the resolution of these ties in the reversed sample caused two extra sequences to pass the similarity threshold in HMMER before no matches were found. It was not specifically checked in the experiment that the sequences were processed in exactly the same order for the forward and reversed samples.

4.3.5 An alternative quality measure

Alignment quality has been measured so far using the TC score calculated on the reference sequences’ core columns. The HMM bitscore provides an alternative means of assessing alignment quality. This approach uses all sequences in an alignment, rather than relying on a small number of reference sequences. Because of this, it avoids the potential for the reference sequences being non-representative of the remaining sequences in an alignment, and also the position of the reference sequences in the overall alignment process.

In this approach to assessing an alignment, the alignment that has been produced is converted into an HMM profile, each of the unaligned sequences are scored against this profile, and the average bitscore of all the sequences is calculated. This average indicates how well the alignment “represents” the input sequences, with better alignments having higher average bitscores. While this approach does not have the weight of more formal benchmarking systems, it is straightforward and can be used with any set of sequences.

To see how well this approach works in practice, one dynamic ordering alignment of 1,000 sequences produced by Clustal Omega, Mafft (FFT-NS-2) and Muscle (2 iterations) was selected for 12 HomFam families. These families were chosen to give a wide range of sequence lengths and hence HMM bitscores. Using the unaligned sequences in each of these 12 cases, a chained guide tree was created with the first two sequences from the dynamic ordering being innermost in the tree and hence aligned first, and the remaining 998 sequences randomly allocated to the leaves of the rest of the guide tree. The sequences were aligned using this chained guide tree, the unaligned sequences scored against the profile HMM of the alignment, and the average bitscore calculated. This process was repeated 100 times, and the results are presented in Figure 4.8. The red dot
represents the average bitscore for the dynamic ordering alignments. The black points representing the randomly ordered sequences were “staggered” from left to right to prevent overlap, and their position along the x-axis is unimportant.

From the results it appears that dynamic ordering works quite well for Clustal Omega, and also to a lesser extent for Muscle. The results are mixed for Mafft, with the approach working for some families but performing poorly for others. The results do show a similarity with those using the TC scores presented in Figure 4.6 above, with dynamic ordering working best for Clustal Omega and worst for Mafft.

4.4 Run-Time Performance

Computing times for the three different alignment programs with default options and dynamic ordering are given in Figure 4.9. All alignment programs were run on a single core of a 2.4 GHz AMD Bulldozer processor running 64-bit Ubuntu 14.04. While the server in question was not exclusively available, any timing runs were conducted when the computer utilisation was less than 50%. Dynamic ordering is consistently 2–3 orders of magnitude slower than the default alignment programs, making the approach in its current implementation infeasible for anything more than a few hundreds of sequences.

Figure 4.10 breaks down the overall elapsed time in Figure 4.9 into the time taken for each step in the alignment process. By far the most computationally expensive step is the initial all-against-all sequence comparison to determine which two sequences are selected to create the initial alignment. This involves firstly creating an HMM profile for each individual sequence and then scoring all sequences against the concatenated profiles. The first step is trivial to parallelise in a bash script — divide all sequences up into \( n \) groups, where \( n \) is the number of cpu threads available, and the HMM profiles for the sequences in each of these groups can be constructed in parallel. When all groups are completed, scoring each sequence against the different profiles can use the multi-threading in HMMER. However, this tends to reach an upper cpu utilisation limit of 400–500% even when running with 8, 12 or 16 threads. Instead, it is more efficient to score each of the \( n \) groups of sequences against the set of HMM profiles in parallel, with each thread returning its best scoring pair of sequences. The highest scoring pair of sequences across all threads can then be easily selected. Figure 4.11 shows the elapsed times of this method for different numbers of threads and sequences. As there is one alignment of just two sequences in this step, with the rest
Figure 4.8: Bitscores for dynamic ordering (red) and 100 random orderings (black) for 12 HomFam families aligned with Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations).
of the time taken by HMMER and file manipulation, the results are just shown for Clustal Omega.

After the all-against-all sequence comparison, the remaining steps in the alignment process require a mixture of HMMER to create the HMM profile and to score the remaining sequences against it, the alignment program itself and bash scripting to select the most similar sequence and extract it from the unaligned sequences. HMMER, Clustal Omega and Mafft are all multi-threaded, and each can be allocated the available number of cpu threads. Figure 4.12 shows the elapsed time for one incremental step in the alignment process for the different alignment programs and various numbers of sequences. Given the inherently sequential nature of the processing, making additional cpu threads available has little impact on the time taken, and may slow down the programs due to overthreading. The scoring of the individual sequences against the HMM profile could also be parallelised in a bash script, as with the all-against-all scoring above. This was not implemented in order to show the efficacy of the multi-threading in the various programs.

Although the most similar sequence is selected at each incremental step of the dynamic ordering approach, all the remaining sequences are ranked in order of similarity. Hence, all sequences except the most similar sequence are the current indications of where they will occur in the dynamic
Figure 4.10: Elapsed times for each step in the alignment process for a given number of Cytochrome P450 sequences on a single thread using dynamic ordering aligned with Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations). The first step includes the all-against-all sequence comparison.

Figure 4.11: Elapsed times for the all-against-all sequence comparison for a given number of Cytochrome P450 sequences with various numbers of threads, and the alignment of the top two scoring sequences using Clustal Omega.
ordering. Naturally, such predictions further out will be less likely as the alignment may have changed considerably by the time that alignment step has been reached. However, the more immediate indications may be sufficiently accurate that it is not necessary to re-determine the similarity ordering after each sequence is aligned. To test this, Figure 4.13 shows the first 4 sequences after the most similar sequence at each alignment step for 100 and 500 sequences randomly selected from three protein families aligned with Clustal Omega. Each sequence is converted into its actual alignment position (y-axis), so that perfectly accurate predictions will fall on the diagonal line.

For all proteins and sizes, the initial predictions were quite accurate. As the alignments grew this accuracy decreased considerably, but towards the end of the alignment the accuracy became more stable particularly with the larger dataset, so it appears that it may not be necessary to recalculate the sequence similarities in the latter stages of an alignment. However, the protein with the most accurate predictions at the end of the larger alignment, Cytochrome P450, was the protein with the least accurate predictions at the end of the smaller alignment. The converse situation is also true: the protein with the most accurate predictions at the end of the smaller alignment, subtilase, was the protein with the least accurate predictions at the end of the larger alignment. Hence, it is not clear
Figure 4.13: Predictions of the next 5 sequences at each alignment step for (top) 100 sequences and (bottom) 500 sequences randomly selected from ace (alpha beta-hydrolase), p450 (Cytochrome P450) and subt (subtilase) using Clustal Omega, Mafft (FFT-NS-2 algorithm) and Muscle (2 iterations).
which sequence predictions could be relied on instead of re-determining the most similar sequence at each alignment step. Nor is it clear what impact the use of predictions might have on the alignment quality. Unfortunately both of these questions are beyond the scope of this chapter.

Finally, the last performance improvement was mentioned in the Chapter Introduction, namely that the proposed approach should run considerably faster if implemented in an integrated program, rather than in bash scripts with considerable amounts of file manipulation as at present. However, this should be considered as the last option, once it is known what other performance improvements should be implemented and when the effort required to develop such a program has been justified.

4.5 Discussion

This chapter set out to address two questions raised by the work presented in the preceding chapters. Firstly, can an ordering of the leaves on chained guide trees other than random produce higher quality alignments. Secondly, can the dependence of an alignment on the ordering of input sequences be avoided. In both cases, the underlying issue was deemed to be the $k$-tuple score used to measure the degree of similarity between two sequences. This score is coarse-grained and gives large numbers of tied $k$-tuple scores with large alignments. This results in order optimisations based on these scores, or the generated alignments themselves, being determined by the order in which the sequences are listed in the input sequence file.

An alternative similarity measure, the HMM bitscore, was used in the dynamic ordering method proposed in this chapter. This has the numerous advantages of Hidden Markov Models, including position-specific scoring, as well as being a more fine-grained similarity measure. While the actual bitscore is a continuous variable, the implementation described in this chapter uses the bitscore to one decimal place reported by HMMER. Hence, this similarity measure still has a finite number of potential values, so an element of instability is still possible. However, as dynamic ordering essentially mimics a chained guide tree topology, tied bitscores may change the ordering of sequences in adjacent alignment steps. This local reordering of sequences should have a lesser effect on the overall alignment produced, in comparison to segregating the sequences into different branches of the guide tree if a more balanced topology was used.

HMM bitscores may also be calculated between a sequence and a profile as between two sequences. Because of the sensitivity of HMMs, calculating the bitscores between the unaligned sequences and the growing alignment allows the “most similar sequence” to be determined after
every alignment step. The effects of the changing alignment on determining sequence similarity were originally identified by Wheeler and Kececioglu (2007) and can be seen here in Figure 4.2. The idea of aligning the most similar sequences first dates back to Hogeweg and Hesper (1984), and while this is the rationale for creating a guide tree in a progressive alignment, all of the more popular progressive alignment programs rely on comparison of sequences made before any have been aligned.

When comparing their relative performance on both the BAiLBASE and HomFam benchmark datasets, dynamic ordering does considerably better than using random chained guide trees in Clustal Omega and Muscle, although the performance of random chained guide trees tends to improve with larger sample sizes. It is possible that with even larger datasets that random chained guide trees would, on average, produce higher quality alignments than dynamic ordering. Unfortunately the data is not available at this time. With Mafft, the results are more mixed. Dynamic ordering performs better than random chained trees in four out of the 6 BAiLBASE reference sets, and the results are neutral in the other two reference sets. On the HomFam datasets, the performance of random chained guide trees increases while that of dynamic ordering decreases from about 250 sequences upwards. However, the results for the majority of HomFam families are neutral.

The results for the default alignment options compared to dynamic ordering show an interesting effect when measured using BAiLBASE and HomFam. On BAiLBASE, the default alignments of all three alignment programs score better than dynamic ordering in alignments with divergent subfamilies, and dynamic ordering scores better than default in reference sets with at least four equidistant sequences. The results are neutral for all alignment programs in the other three reference sets. In HomFam, for both Clustal Omega (particularly) and Muscle, dynamic ordering performs better than the default options from about 250 sequences onwards. For the smaller datasets, the results are more neutral, with the largest number of protein families having no significant differences in performance. Given that the BAiLBASE reference sets are quite small, with the largest having 142 sequences, it seems that the impact of dynamic ordering becomes more apparent after about 250 sequences. The default alignment options are generally better than dynamic ordering for Mafft, although the trend is for the percentage of families scoring better to slowly increase with dynamic ordering and slowly decrease with the default alignment options.

Using the HMM bitscore as a measure of sequence similarity has greatly reduced the variation between alignments produced with different ordering of the same set of sequences, as was shown in
Figure 4.7. However, the details given for the fabp protein family highlight one of the main aspects of the dynamic ordering approach. With the fabp family, selecting a different sequence when the next most similar sequence could not be found automatically resulted in a considerably different alignment being produced as measured by the alignments’ TC scores. The dynamic ordering approach, like progressive alignment, is a greedy algorithm that tries to determine the best alignment ordering but makes no attempt to correct errors made in previous alignment steps. Starting the alignment process with the two most similar sequences may not be the best alignment strategy. Once the two initial sequences are aligned, the ordering of the remaining sequences follows, but other options for selecting the initial sequences may result in a better overall alignment. This approach was adopted here, based on the premise that aligning the most similar sequences first yields the best alignment. That may not be the case in all scenarios. An alternative strategy might include finding the two most “central” sequences, e.g. those with the highest average bitscore when measured against all other sequences. Alternatively, using the two longest sequences might be useful as these may provide a more representative alignment onto which the other sequences could then be added. Not filtering the input data to ensure all sequences are approximately the same length would favour the selection of longer sequences initially, as these would tend to give higher bitscores. Without further analysis it is not clear which alternative strategy, if any, would give the best alignments.
Chapter 5

Summary and Future Work

The focus of this thesis has been on the use of guide trees in large-scale progressive protein multiple sequence alignment algorithms, and in particular their impact on the quality of the alignments produced by these algorithms. Although the progressive multiple sequence alignment approach was initially proposed 30 years ago, alignment algorithm development is still an active research area. This is partly due to its widespread use as a first step in the comparative analysis of homologous genes or proteins, and also due to the new challenges being addressed, many of which are related to the increased availability of sequence information over the years.

The growth in the number of protein sequences has identified a number of issues with the alignment algorithms that were proposed when the field was still in its infancy. In Chapter 1 the falloff in alignment quality, as measured using a structure-based benchmark, as the number of sequences to be aligned increased was highlighted, and this served as a primary motivation for the work presented here. In Chapter 2, the guide tree topology was shown to have an impact on alignment quality, again as measured using a structure-based benchmark, particularly as the number of sequences reached about 1,000 sequences. Perhaps most contentious was the finding that the quality results from randomly ordering the sequences on a chained guide tree were as good as those when the ordering was optimised based on the similarity measure used to construct the default guide trees. In hindsight, the latter part of this finding was not fully recognised at the time, and the message understood by many was that randomly ordered chained guide trees gave results comparable with optimised guide trees. It was only later, when the results presented in Chapter 3 became available, that the lack of information in the pairwise sequence similarity measure was recognised.

Subsequent to the publication of these findings, other studies have come to similar conclusions.
Sievers et al. (2014) examined all possible guide-tree topologies for small numbers of protein sequences, and found that on average chained guide trees produce better results than balanced guide trees. While they could not confirm that random chained guide trees are better than the default trees produced by the most common alignment programs, they did find that the superiority of default guide trees over chained ones decreased as the number of sequences being aligned increased from 4 to 8 and then to 16 sequences. This lead them to suspect that, beyond a certain number of sequences, the alignment quality obtained using random chained guide trees would surpass that of the default distance-based guide trees.

Fox et al. (2016) used de novo protein structure predictions to create a benchmark, ContTest, to measure the quality of alignments of very large numbers of protein sequences produced by many of the more popular protein multiple sequence alignment programs. They also found that chained guide trees increase the accuracy of large-scale alignments. These results were confirmed recently by Yamada et al. (2016), who found that the latest version of Mafft performed better using random chained guide trees on ContTest than the default trees. However, they also found that default trees performed better than random chained guide trees when quality was measured using the HomFam structural benchmark.

Chapter 3 showed that there was an inherent instability in the most popular progressive alignment algorithms, as these relied on a $k$-tuple based score to determine the similarity between sequences. As the number of sequences increased, large numbers of tied $k$-tuple scores were being generated. Without any other information to cluster the sequences when creating the guide tree, the alignment algorithms relied on a sequence's position in the input file. Hence, changing a sequence's position relative to other sequence pairs with the same $k$-tuple scores could result in a different guide tree being formed, and hence a different alignment being produced. This has significant ramifications for the reproducibility of research that includes a multiple sequence alignment, as most people documenting their research would not specify the order in which the sequences were listed in the input file to the alignment program.

Finally Chapter 4 attempted to draw together some of the findings of the previous two chapters to form an integrated research programme for the whole of this thesis. While random ordering was shown in Chapter 2 to produce alignments with similar quality as those with an optimised ordering, all the optimisations were based on the $k$-tuple similarity score. This was shown in Chapter 3 to have poor discriminatory power between sequences when the number of sequences became large.
Hence, by using a more fine-grained measure of sequence similarity, it was hoped that a better ordering of sequences than random in chained guide trees could be found. In addition, such a similarity measure may also eliminate the dependence an alignment has on the ordering of sequences in the input file. The proposed similarity measure, the HMM bitscore, together with the continuous updating of the next “most similar sequence” was a partial success. The dependence on sequence order was removed in virtually all cases, and the means to tackle the problem in the remaining situations was proposed. The quality of the alignments produced with this dynamic ordering approach was better than that of the default alignment options and random chained guide trees in most larger alignments in Clustal Omega, but the results were more mixed with Mafft. In addition, the time requirements for dynamic ordering were far worse than for the default alignment options, although the performance was easily improved by using additional cpu threads.

This point is important. Alignment programs allow alignments of many sequences to be made quickly, even on personal computers, as Higgins and Sharp (1988) pointed out a number of years ago. While the definition of what constitutes many sequences and the size of a typical alignment may have increased over the years, for most users their preferred multiple sequence alignment program performs adequately for their needs. This will obviously depend on the purpose for which the multiple sequence alignment is being created, but for many users, a quickly generated alignment of a small number of sequences is sufficient. Biological workflows are often messy and produce conflicting results and most biologists have learned to work around the issues, e.g. using a particular alignment program or parameter options with specific data, or manually intervening to adjust the alignment results prior to downstream analysis (Morrison, 2009). This is perhaps the bioinformatics equivalent of Simon’s (1956) “satisficing”, a decision-making strategy which selects the first alternative that meets a certain acceptability threshold without it necessarily being the optimal solution. The term is a combination of “satisfy” and “suffice”, and is often used in situations where an optimal solution cannot easily be determined. For most biologists, the optimal alignment solution is not known, and they rely on the multiple sequence alignment program to determine this, or an adequate alternative, for them.

It is for this reason that benchmarking the different alignment programs is so important. As was discussed in Chapter 1, there are numerous benchmarks available for assessing the relative merits of the alignment programs for different types of workflows and analysis. However, more can be done, particularly when dealing with the instability in alignment algorithms identified in Chapter 3.
Multiple sequence alignment programs should issue a warning when duplicate distance measures are encountered when creating the guide tree, to indicate to the user that an alternative alignment will probably be generated if the order of the sequences in the input file is changed. While any alignment generated is just one of a multitude of potential alignments, most users of alignment programs do not expect their results to be dependent on how they constructed their data files. In addition, the benchmarking of any alignment program should also report on any difference in performance when the sequence order is reversed. The instability in alignment algorithms is primarily an issue with large-scale alignments, but unfortunately many new uses of alignments, e.g. protein structure prediction, require such large alignments.

The dynamic ordering approach proposed in Chapter 4 does address the instability issue by using both a more fine-grained similarity measure and, implicitly, chained guide trees. However, the approach should be seen as a work-in-progress with many details still to be resolved. The choice of the initial sequences in the alignment process is crucial in determining the overall alignment that is generated. Other options besides the two most similar sequences should be considered, as the two most similar sequences depend on a local comparison and may not be representative of the other sequences in the alignment. The most “central” or the two longest sequences were suggested previously, and other options are also available, including the sequences with the lowest variability in bitscores when compared with all other sequences. Whatever method is chosen, however, should be quick and easily implemented.

It would also be interesting to assess dynamic ordering with larger HomFam families. After deduplication and filtering, there are 16 HomFam families with 5,000 or more sequences, and only 7 with more than 10,000 sequences. Because of time and computational constraints, it was not feasible to compare the performance of dynamic ordering, random chained guide trees or the default options for these families. Random chained trees showed their main effect with larger alignments of about 1,000 sequences upwards. Dynamic ordering appears to have a positive impact with a smaller numbers of sequences, in the order of 250 to 500 sequences. It is not clear with the data to date whether random chained or dynamic ordering will have the greater effect on larger numbers of sequences.

Yamada et al. (2016) reported mixed results for the latest version of Mafft using random chained guide trees as measured with HomFam and ContTest. It’s not clear why HomFam and ContTest gave different results, but it would be interesting to use ContTest to assess the quality of alignments.
produced by dynamic ordering. As ContTest uses all sequences in an alignment, it avoids the issue HomFam has with the location of the reference sequences in the overall alignment. However, before doing so, it would also be worth seeing if ContTest reported different quality scores when the order of sequences is changed in the input file. It might also be informative to try to reconcile the Mafft results reported earlier with those of Yamada et al. by reporting absolute TC scores rather than just inequality as in the Wilcoxon statistical test, and also by protein family.

While dynamic ordering’s time performance was quite poor, there are a number of options that could be considered to improve this. Performance could be greatly reduced by replacing the existing bash script solution with an integrated program. In addition, HMMER is an exceptionally elegant package, but its multi-threading can be improved. Instead of trying to parallelise the processing of a particular sequence against the HMM profile to determine its bitscore, a much greater performance improvement can be achieved by splitting the input file across the available threads, as was done in all-against-all comparison in Section 4.4. In comparison, in the incremental alignment step when all remaining sequences are scored against the profile of the partial alignment, there is no reduction in elapsed time by allocating more CPU threads to the process.

Finally, it is worth restating that progressive multiple sequence alignment algorithms are heuristic, and cannot generate optimal alignments. Their widespread use is due to their ease of use and generation of good quality alignments, requiring relatively modest computing resources. It is important that these aspects are not lost as new approaches are developed to address the existing and future challenges.


Russell, R. B. and Barton, G. J. (1992). Multiple protein sequence alignment from tertiary structure


Appendices
Appendix A

Data files, Utility programs and Scripts

A package of utility programs, scripts and data files referenced throughout this work is available for

It contains 3 separate folders, one each for Chapters 2, 3 and 4. Each folder contains a number
of data files, programs and scripts, and a README file specific to that folder which includes descrip-
tions of each of the files in that folder. Some of the programs, e.g. reshuffleSeq.c are common to
each Chapter but are included in each folder and described in each of the 3 README files.