COMPARISON OF MULTIPLE SEQUENCE ALIGNMENT PROGRAMS IN PRACTISE

.

by

Nguyen Thu Hang

A Thesis Submitted to

The Bioinformatics Research Center (BiRC) University of Århus

In Partial Fulfillment of the

Requirements for the degree of

MASTER OF BIOINFORMATICS AT UNIVERSITY OF ÅRHUS

Approved:

____

Christian N. S. Pedersen, Thesis Adviser

The Bioinformatics Research Center (BiRC)

University of Århus

Oct, 2008

AARHUS UNIVERSITY
BIOINFORMATICS RESEARCH CENTER
C.F.MØLLERS ALLE, BUILDING 1110

Title:

COMPARISON OF MULTIPLE SEQUENCE ALIGNMENT PROGRAMS IN PRACTISE

Synopsis:
The thesis was intended to explore and verify the statements about the application of some of the most outstanding multiple sequence alignment (MSA) programs in real practise. Based on the documentations and the papers referenced by these programs, the thesis try to implement a unified experiment to test these programs with the same conditions and then compare the results in both accuracy and running time measure. Using the benefits of the open sources of these MSA programs, python programming langue and Linux batch programming, the thesis has processed a full test of the programs on the BaliBASE alignment reference database, giving a deep insight of the performance of each program in practise. The experiment can be also applied for further MSA programs comparisons, requiring the same structure of database and in/output formats.

Student name:
Nguyen Thu Hang

Number of page:
70

Suppervisor:
Christian N. S. Pedersen

CONTENTS

ACKNOWLEDGMENT

I would like to say many thanks to my supervisor Christian N. S. Pedersen for the unconditional support when I do this thesis. Thanks for all his advises he has been giving which has turned into a big motivation for me.

1. Introduction

Multiple sequence alignment (MSA) has become widely used in many different areas in bioinformatics from finding sequences family, detecting structural homologies of protein/DNA sequences, predicting functions of protein/DNA sequences to predict patient's diseases by comparing DNAs of patients in disease discovery. MSA is the major interest in bioinformatics and the fundamental step for almost other molecular biology analyses. MSA is the most natural way to see the relation between sequences by making an alignment between the primary sequences so that identical or similar residues will be aligned in columns. That is why this method is so called multiple sequence alignment (MSA).

Definition 1.1: **Multiple Sequence Alignment (MSA)** *is an alignment of more than two sequences*

1.1 Intentions

Nowadays, the development of genome projects and proteome projects give researchers the chance to access a huge data of DNA/protein sequences. Having a good multiple sequence alignment will help biologists to find the answer for a lot of problems. Most interesting problems evolve an extremely long sequences which are various in both length and residues that can only be solved by an appropriate method or a programming technique. Choosing a good MSA to use is also important in research as well as having a method to evaluate different MSA algorithms and software applications.

In this thesis we would like to present 3 most popular multiple alignment methods by reviewing their methodology and evaluating their quality by a unified experiment. The three MSA programs are

- CLUSTALW
- T-COFFEE
- MUSCLE

The experiment will be processed on *BAliBASE* , one of the best *database of manually-refined multiple sequence alignments specifically designed for the evaluation and comparison of multiple sequence alignment programs*[BAliBASE]. We will run a full test for every single test case, collecting the results; verifying them and then based on the results we will compare the performance of three programs in both accuracy and speed. We expect to see the patterns of the accuracy and speed as we expected from the referenced papers of each programs and review the advantages and disadvantages of each program. If there is some peak or bottom of the pattern which represent a sudden change in the performance of a program, then we can track down the test case and the corresponding MSA values.

.

2. Basic of multiple alignment algorithm

In order to get a deep understanding about different MSA programs and their methodologies, we ought to get a basic idea about how to produce multiple alignment algorithm in basic. So in this chapter we will formalize the basic MSA algorithm and introduce the basic exponential time MSA method.

2.1 Sequences and comparison between sequences

2.1.1 Sequence

MSA algorithm is the algorithm to produce MSA from a set of input sequences. So first we will define a sequence as below:

Definition 2.1.1: A sequence is an ordered string s of symbols over a set of alphabet S

In bioinformatics, a DNA sequence (DNA secondary structure sequence) is a sequence is the string over the set S of 4 symbols of nucleotides adenine (A), cytosine (C), guanine (G) and thymine (T): *S = {A,C,G,T}*, written as *s = ATTTCTGTAA*.
Similarly, a protein sequence s’ is a string over the set of 22 symbols of amino acids
S = {G, P, A, V, L, I, M, C, F, Y, W, H, K, R, Q, N, E, D, S, T}, for instance s’ can be:
s’ = *GKGDPKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKT*

2.1.2 Comparing multiple sequences and MSA algorithm

The comparison between sequences here is the comparison of the sequences over the same alphabet. It means we can not align a DNA sequence with a protein sequence, instead we should covert DNA sequence to protein sequence or vice versa and then align them together.

Definition 2.1.2.1: Mutliple sequence alginment

Let s_1, …s_k be a set of input sequences over the same alphabet S. Then we will call $R_{1,\ldots,}R_k$ be the set of alignment rows which are obtained by inserting spaces (gaps) "-" into sequence s_1, …s_k in such a way as to make them all of the same length l. Since l is the common length of all sequences after being inserted with gaps, a **multiple alignment M** *is a matrix of size k x l obtained by putting k rows $R_{1,\ldots,}R_k$ row by row*

$$M = \begin{bmatrix} R_1 \\ \ldots \\ R_k \end{bmatrix}$$

To compute a good MSA we need to find a method to evaluate an alignment so that we can find the best one. If we can find a score-function that give an alignment a score indicating the quality of the MSA, then we can compute a good alignment by computing the alignment with the best score value. The goal of a MSA algorithm is to use a *score-function* and find the MSA that maximize that score. Or say in another way, given a score function, we can define a multiple alignment algorithm is the algorithm to find the maximum score of a given alignment *M* as below:

Definition 2.1.2.2: Given a SCORE-function: SCORE(M) of a multiple alignment M, **a Multiple sequence alignment algorithm** is to find $M^* = \arg\max_M SCORE(M)$

2.1.3 The Sum of Pairs (SP) score

We need a good score-function. One of the most popular, simple and often used measure for the quality of multiple alignment is so called as the Sum of Pairs score *SP score.*

Definition 2.1.3.1: Given a multiple alignment *M*, **an induced pairwise alignment** of two rows R_i and R_j: *induced_pairwise(R_i, R_j)* is the multiple alignment M restricted to R_i

and R_j and it is obtained from M by deleting all rows except the two rows R_i and R_j. In an induced pairwise alignment, any column that contains only gaps will be removed.

Definition 2.1.3.2: The score of an induced pairwise alignment: *Pairscore(R_i, R_j)* can be any chosen scoring scheme for pairwise alignment in the standard maner. The *Pairscore* is *column-based* if the scoring scheme is the global pairwise alignment with linear gapcost.

Definition 2.1.3.3: The Sum of Pairs (SP) score of a multiple alignment M of *k* sequences is the sum of the scores of all its induced pairwise alignments:

$$SP(M) = \sum_{1 \le i, j \le k} induced\_pairscore(R_i, R_j)$$

2.2 The exact exponential time SP alignment algorithm

The exact SP multiple alignment algorithm can be calculated by dynamic programming. The pseudo-code of the exponential time algorithm for the case of k=3 is written in table 2.2.1. The development of the algorithm for k>3 is straight forward [HollyGrail].

Pseudo code notation:

- S_1, S_2 and S_3 denote three sequences of lengths n_1, n_2 and n_3 respectively
- D(i,j,k) is the optimal SP score for aligning S_1[1..i], S_1[1..j], S_1[1..k].
- $D_{1,2}$(i,j) denotes the pairwise distance between (S_1[1..i] and S_2[1..j]). Similarly, $D_{1,3}$(i,k) and $D_{2,3}$(j,k) denote the pairwise distances of (S_1[1..i] and S_3[1..k]) and ((S_2[1..j] and S_3[1..k])) respectively.
- *smatch, smis,* and *sspace* are the score for a match, mismatch, or gap (space) respectively.

The exact SP alignment algorithm:

```
# Recurrences for a nonboundary cell (i,j)
1:   for i:=1 to $n_1$ do
         for i:=1 to $n_1$ do
             for i:=1 to $n_1$ do
                 Begin
5:               if ($S_1$(i) = $S_2$(j)) then cij := smatch
                 else cij := smis;
                 if ($S_1$(i) = $S_3$(k)) then cik := smatch
                 else cik := smis;
                 if ($S_2$(j) = $S_3$(k) then cjk := smatch
10:              else cjk := smis;
11:              d1 := D(i-1, j-1, k-1) + cij + cik + cjk;
12:              d2 := D(i-1, j-1, k) + cij + 2*sspace;
13:              d3 := D(i-1, j, k-1) + cik + 2*sspace;
14:              d4 := D(i, j-1, k-1) + cjk + 2*sspace;
15:              d5 := D(i-1, j , k) + 2*sspace;
16:              d6 := D(i, j-1, k ) + 2*sspace;
17:              d7 := D(i, j, k-1) + 2*sspace;
                 D(i,j,k) := Min (d1, d2, d3, d4, d5, d6, d7)
                 end;

# for a boundary cell
19:      D(i ,j, 0)  =  $D_{1,2}$(i, j) + (i + j)*sspace,
         D(i, 0, k) =  $D_{1,3}$(i, k) + (i + k)*sspace,
         D(0, j, k) =  $D_{2,3}$(j, k) + (j + k)*sspace,
22:      D(0,0,0) = 0
```

Table 2.2.1: the pseudo-code of exact SP MSA algorithm for the case of k=3

Pseudo-code explanation: The main idea of the peuso-code is using dynamic programming algorithm to compute optimal SP alignment of three sequences $S_1[1 \dots n_1]$, $S_2[1 \dots n_2]$ and $S_3[1 \dots n_3]$. The algorithm will use an 3 dimensions array (or a 3D cube) D of size $(n_1+1) \times (n_2+1) \times (n_3+1)$ in which each cell $D(i, j, k)$ is the optimal SP score of aligning the prefixes $S_1[1 \dots i]$, $S_2[1 \dots j]$ and $S_3[1 \dots k]$. The value of an inner cells, i.e. $D(i, j, k)$ with $i, j, k > 0$, depends on seven adjacent cells $D(i-1, j-1, k-1)$, $D(i-1, j-1, k)$, $D(i-1, j, k-1)$, $D(i, j-1, k-1)$, $D(i-1, j, k)$, $D(i, j-1, k$ and $D(i, j, k-1)$ and the optimum for the cell $D(i, j, k)$ will be the optimum, which here is the *minimum*, of seven cases evolved seven adjacent cells but they are actually belong to one of these groups of cases with the reference lines from the pseudo-code as below:

- Line 11: first, an optimal alignment of the three prefixes $S_1[1 \dots i]$, $S_2[1 \dots j]$ and $S_3[1 \dots k]$ can be retrieved by aligning the last chars of each prefixes plus the previous-computed value of $D(i-1, j-1, k-1)$:

  $D(i-1, j-1, k-1) + s(S_1[i], S_2[j]) + s(S_1[i], S_3[k]) + s(S_2[j], S_3[k])$

  where s is the score for match, mismatch or space

- Line 12 – 14: second group is for the cases that any of the three sequences ends with a space aligned with the last chars of the other two, giving a score:

  $D(i-1, j-1, k) + s(S_1[i], S_2[j]) + 2*sspace$

  (if "-" was inserted at the end of S_3;; the cases for S_1, S_2 are similar)
- Line Finally, any two of the sequences end with a space aligned with that char of the third sequence:

  $D(i-1, j, k) + 2*sspace$

  (if S_3 is the third sequence, the cases for S_1, S_2 are similar)

The value for a boundary cell is applied straightforward as in lines 19 – 20.

Each cell of the cube D can be filled in constant time, so the total time of the recurrences is $O(n_1 n_2 n_3)$. Expand for the cases if there are k sequences of length n, the dynamic programming will take $\Theta(n^k)$.

3. Review of existing programs

3.1 Difference approaches of MSA programs

Even though there have been many algorithms and software programs implemented to produce multiple sequence alignments of protein and DNA sequences but there are only several main approaches. The basic approach, also called as common heuristic alternative, seeks a multiple alignment that optimizes the sum of pairs (SP) score. This approach is easy to implement and produce high quality alignment but the running time is NP complete. If we use dynamic programming the smallest time and space complexity for the heuristic approach we can get is $O(L^N)$ in sequence length L and the number of sequences N which means we can run on very small N in practice [Waterman].

Another mathematical approach is so called probabilistic and stochastic methods which using probability in producing MSA. Hidden Markov Models (HMMs) is the popular example of this approach, in which MSA problem is modeled as probabilistic models where all possible combination of gaps, matches, mismatches are assigned with probabilities and the algorithm will find the most likely MSA or set of possible MSAs. Although HMMs offer significant improvements in computational speed, especially for homologous sequences, but they do not produce the same solution every time running and can not be guaranteed to converge to an optimal alignment [Edgar]. Another example of stochastic approach is genetic algorithms and Simulated annealing. Genetic algorithms attempt to break a series of possible MSAs into fragments and repeatedly rearranging them. Simulated annealing method can use an existing MSA and refines it by a series of rearrangements it can find more optimal alignment regions than the existing MSA already occupies and maximize an objective function like the SP score function [Goldberg]. In general, stochastic methods have been developed relatively recently and they are more complex than using more common progressive methods and have not been widely adopted.

The most wide used approach is the progressive approach which also called as hierarchical or tree methods. The strategy of making MSA is aligning the most similar sequences and then adding successively less related sequences or groups of sequences to the alignment until the MSA contains all the sequences. Progressive approach constructs phylogenetic tree to guide the order of aligning sequences or groups of sequences into the MSA so using the technique the programs actually produces both phylogenetic tree and MSA. Many popular multiple alignment softwares have been using progressive approach. ClustalW is known to be the fast and light program that can align a big number of sequences in a micro computer and has been adopted to different variations of clustalW for multiple sequence alignment, phylogenetic tree construction or even as input for other programs such as T-Coffee (Tree-based Consistency Objective Function For alignment Evaluation) or protein structure prediction etc.

Another famous progressive multiple sequence alignment program is T-Coffee which provide a dramatic improvement in accuracy so that even it is slower than commonly used alternatives but as a reward T-Coffee is considered to be the recommendation when you want to produce high accurate MSAs. T-coffee not only uses the popular progressive approach but also improve the technique to avoid the most serious pitfalls caused by the greedy nature of this algorithm [T-Coffee]. Before constructing the phylogenetic tree, T-Coffee has a special way to calculate pairwise alignments so that each sequence will be considered not only on how they align to each other to each other but also on how all of the sequences align with each other. It is done by combining the direct alignment of the pair with indirect alignments that aligns each sequence of the pair to a third sequence. Beside, T-Coffee takes into account the information of both global and local alignments as well as using advantages of other global and local alignment programs by using ClutsalW and the local alignment program LALIGN as input. In the last versions, 3D-Tcofee can use structure information from Protein Data Bank.

Even thought progressive approach is the most widely used approach to multiple sequence alignments but it still has some weak points. One of the main weak points is the heavy dependence on the accuracy of the initial pair-wise alignments. With the attempt to improve the progressive methods, another approach so called Iterative methods has been developed. The strategy is similarly to progressive methods but repeatedly realign the initial sequences as well as adding new sequences to the growing MSA. Various iterative alignment software adopt the strategy in various ways: for instance PRRN/PRRP uses a hill-climbing algorithm which optimizes its MSA alignment score and iteratively corrects both alignment weights and locally divergent or "gappy" regions of the growing MSA; DIALIGN takes focus narrowly on local alignments between sub-segments or sequence motifs without introducing a gap penalty; CHAOS/DIALIGN uses fast local alignments as "seeds" for a slower global-alignment etc. Among these programs, the most outstanding popular iteration-based method is MUSCLE (MUltiple SequenCe alignment by Log-Expectation) which is tested to be the fastest of the tested methods for large numbers of sequences. It has the advantage of using the comprehensive improvement strategy and more accurate distance measure which can be updated in each stage of the iteration [MUSCLE].

Each of the programs has it owns advantages and in some specific situation, one can be better than the other. In this thesis, we would like to choose three of the most popular softwares: ClusalW, T-Coffee and MUSCLE and verify the qualities of the software in practice. In the next chapter we would like describe in details about the algorithms and implementations of these program their common discussion about their qualities that researchers have been noticed in practices.

3.2 T-coffee (Tree-based consistency objective function for alignment Evaluation)

T-Coffee is the short name which stands for Tree-based consistency objective function for alignment Evaluation. The program is broadly based on the progressive approach to multiple alignment but the algorithm has been improved and developed to avoid the serious greedy problem caused by this approach [T-Coffee]. T-Coffee progressive alignment algorithm basically is a circle of 3 main stages: pre-processing a library, pairwise alignments, and refining the library. The refined library will continue guiding the progressive alignment. While running the algorithm to produce the final MSA, T-Coffee produces several intermediate alignments that can be used and the algorithm can also stop at that point. T-coffee uses the primary library of both global and local alignments to measure distance between pairwise sequences or combines multiple alignments from other programs.

The T-coffee algorithm

The process of producing alignment of T-coffee is basically the circle of running of two main procedures: Generating MSAs and Optimization. Generating MSAs is the procedure of getting global and local alignments as input and output some simple flexible multiple alignments that again can be refined and chosen. Optimization procedure is the process of using progressive alignment [ClustalW] to find the multiple alignments that fits the input library the most. The process of the whole T-coffee algorithm will be illustrated as bellows:

Assume we have a set of n sequences to be aligned, so there will be $n(n-1)/2$ pairs of pairwise sequences. For each pair of sequences, T-coffee will run clustalW to produce the global pairwise alignment and Lalign program to produce one or more local alignments and collect these alignments in to a library called primary library of alignments. Each alignment in the library will be presented as a list of pairwise residues so called constraints. Each constraint then will be weighted using a weighting scheme in such a way that *"each constraint receives a weight equal to percent identity within the pairwise alignment it comes from"*.

Because we have run two different programs on each pair of sequences, we will have two libraries of lists of constraints, one for local alignments and one for global alignments. The question now is how to combine these libraries so we will have a united library. Using a simple process with 3 simples rule: duplicated pairs will be merged into a single entry that has a weight equal to the sum of the two weights otherwise a new entry is created. Pairs of residues that did not occur will be weighted as 0. By that way, T-coffee combines the two global and local primary libraries into a merged library with new weights. T-coffee then exams the consistency of each pair of residues with residue pairs from all other sequences. T-coffee takes each aligned residue pair from the library and checks the alignment of the two residues with residues from the remaining sequences. If l is the average sequence length of the sequences then the complexity of building the extended library pairwise is $O(n^3l^2)$ [T-Coffee]. The extended library will be used as input to the optimization procedure.

From the extended library, pairwise alignments scores are pooled out to build a distance matrix between all the sequences. Using this distance matrix, T-coffee will build a phylogenetic tree using the neighbor joining method. This phylogenetic tree will be used to guide the order of building MSAs in such a way that the closest sequences in the tree will be aligned first. Dynamic programming will be used to build the MSA.

To evaluate the quality of T-coffee program, in the paper [T-Coffee], they run the experiment on BaliBase databases and compare the results with the results running by other MSA methods using default parameters. They state that T-coffee shows the highest average accuracy in each BaliBase category among Prrp, Dialign and ClustalW.

3.3 Clustal W

ClustalW is one of the most common used progressive MSA methods because of the advantages of speed and the size of the program. The main idea is beside applying the basic progressive alignment method, ClustalW also uses other tactics such as assigning individual weights to each sequence to down weight near duplicate sequences and up-weight the divergent sequences, position-specific gap penalties an the flexible of choosing substitution matrices for different alignment stages. This section will discuss how ClustalW implement the progressive method and the quality of the alignments it produces.

Assuming if we have *n* sequences as input sequences to be aligned, ClustalW will first follow the basic progressive alignment procedure:

Distance matrix:

- Building a distance matrix from all the pairwise alignments to measure the divergence of each pair of sequences: $n(n-1)/2$ pairs of sequences are aligned separately and the results will be used as pairwise distances. These pairwise distances will be collected to build a matrix of distances with columns and rows are the sequences like in the example of distance matrix in the figure below.
- There are two options to calculate the pairwise distances by using a fast approximation method or using a slower but more accurate full dynamic programming method. This option allows us to align a large numbers of sequences in a microcomputer if we choose the first option or reaching to the best quality of alignment using the second option.

The guide tree

Similar to T-coffee, ClustalW also build the guide tree to guide the order of building the final MSA from the distance matrix. But in ClustalW, the guide tree will also be used to derive a weight for each sequence.

- ClustalW uses the Neighbour Joining method [Neighbor] to build the guide tree in which each branch present a sequence to be aligned and the branch lengths proportional to measured divergence [ClustalW].

- For each sequence, ClustalW will assign a weight so that sequences which have a common branch with other sequences share the weight derived from the shared branch.

Building MSA

The idea of building the final MSA is aligning the calculated pairwise alignments to the MSA so that the MSA contains larger and larger numbers of sequences until it contains all the n sequences. It is not efficient to add the alignments in random order. The best way is aligning the pairwise alignments (sequences) in a sequence so that the group of sequences that are more similar will be aligned first. The guide tree will be used in this step to guide the order of adding alignments so that the sequences with smaller distances will be aligned first by following the guide tree starting from the tip to the root, ClustalW align the sequences into the MSA. In each step of adding more sequence to the MSA, ClustalW uses the dynamic programming method with residue weight matrix and two penalties for opening and extending gaps. In the process of aligning two groups of sequences from 2 alignments, if there has been a gap then it will always be a gap. No gap in older MSAs will be deleted; instead new gap will be introduced in the new MSA with full gap opening and extension penalties. An example of gap introduction is demonstrated in Figure 3.3.2.

In each step, the score of a column in the newer MSA alignment will be the average of all pairwise scores of comparing each pair of residues of the same column in the two older MSAs. If the sequences are weighted then each pairwise score will be multiple with the weights that the 2 residues belong to before taking the average..

	1	2	3	4	5	6
1						
2	0.17					
3	0.59	0.60				
4	0.59	0.59	0.13			
5	0.77	0.77	0.75	0.75		
6	0.81	0.82	0.73	0.74	0.80	
7	0.87	0.86	0.86	0.88	0.93	0.90

Figure 3.3.1: An example of a distance matrix using in ClustalW: all pairs of sequences are aligned separately, using either a fast approximate method or from full dynamic programming alignments with two gap penalties. The scores are converted to distances by dividing to 100 and subtracting from 1.0 to give the number of differences per site [ClustalW].

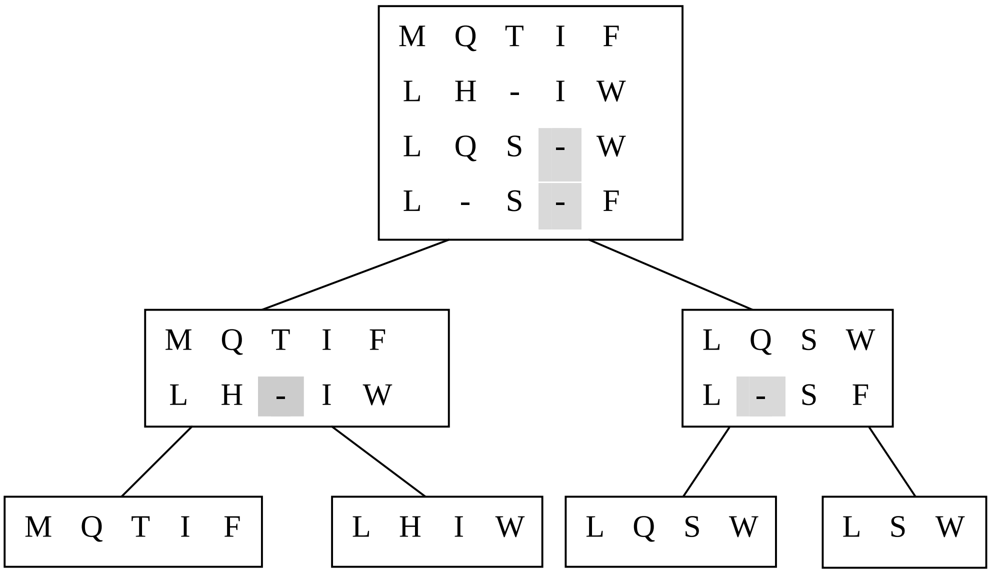

Figure 3.3.2: An example of progressive alignment is built in order and in each step gaps will be introduced to maintain the rule: if there is a gap then it will always be a gap.

3.4 MUSCLE

MUSCLE is considered as a new generation of multiple sequence alignment programs for protein sequences because it has the advantages of both accuracy and speed. Researchers have been compared the speed and the accuracy of MUSCLE with T-Coffee and ClustalW on four test sets of reference alignments: BaliaBAliBASE, SABmark, SMART and PREFAB and they state that MUSCLE achieves the highest or joint higest rank in accuracy and is the fastest method for alignment larger numbers of sequences.

Different from other multiple sequence alignment programs MUSCLE doesn’t apply only heuristic nor only progressive method. Instead MUSCLE considers a computational approach in which multiple alignment problems will be modeled as graphs with probabilities i.e. nodes are the observed sequences and edges are elementary sequence edits with probabilities assigned. Finding the alignment is finding a most probable directed graph and by combining the advantages of progressive alignment and profile alignment. MUSCLE develops a tractable way for finding the graph.

The MUSCLE algorithm has three stages: making a draft progressive alignment, improved progressive and refinement:

The first stage is a straight forward building a progressive alignment with four steps: similarity measure, distance estimate, tree construction and progressive alignment. The similarity measure step begins with the similarity of each pair of sequence is calculated by using either k-mer counting or from pairwise global alignment and collected to build the distance matrix in the next step. Then a guide tree is built from the distance matrix using UPGMA or neighbor-joining. Following the guide tree, a progressive alignment is constructed in the order from the tip to the root similarly to the building MSA step in ClustalW program.

The second stage is the process of improving the progressive alignment including improving the guide tree and building new progressive alignments following the improvement of the guide tree. There are four steps are repeated in the improved progressive stage: similarity measure, tree construction, tree comparison and progressive alignment. In the similarity measure, instead of calculating from each pair of sequences, the similarity of each pair of sequences is computed using fractional identity computed from their mutual alignment in the current MSA. It is because since we already have a MSA, MUSCLE therefore can use the Kimura method which is more accurate but requires an alignment. The Kimura distance for each pair of sequences is computed from the current MSA and collected producing a new distance matrix. A new guide tree is built by clustering the new distance matrix using UPGMA. The old and the new trees are compared to find out the changing branching order of the set of the corresponding nodes. Note that, if stage 2 is repeated and the number of these nodes has not decreased then the iteration of stage 2 will be stopped because at that time the process of improving the tree is considered as converged. Following the new guide tree a new progressive alignment will be built following the new guide tree with optimization. That means new alignments will only be recreated for the set of changed nodes while the rest of the existing alignment is conserved. Say in another way, following the order of the new guide trees from the tip to the root, MUSCLE aligns the existing alignment for which the branching order is unchanged with re-created alignments corresponding to the set of changed nodes and when the root is completed MUSCLE may repeat the second stage if the set of the changed nodes is decreased or go to the third stage: refinement.

Refinement contains fours steps so called as choice of bipartition, profile extraction, re-alignment and accept/reject which are repeated throughout the process, The process uses the new guide tree produced from the second stage and the edges are visited in order of decreasing distance from the root.The first step is removing an edge from the tree, dividing the tree into 2 subtrees and the sequences into two subsets. Then MUSCLE extracts the profile, the multiple alignment of each subset, from the current multiple alignment in which columns with only gaps will be removed. Then the two profiles will be re-aligned yelling a new multiple alignment. The SP score of the new multiple alignment is calculated. If the score is higher then MUSCLE keep the new alignment else the alignment will be discarded. The four steps will be looped until convergence, when all the edges have been visited or it meets a user defined limit.

Discussing about the running time of MUSCLE, there are 3 places corresponding to 3 stages in the algorithms that produce complete multiple alignments. Computing the alignment in the first two stages takes $O(N^2L + NL^2)$ and the refinement process takes $O(N^3L)$ of time [MUSCLE].

4. The BAliBASE Benchmark

4.1 The BAliBASE Benchmark

BAliBASE is the database containing high quality multiple sequence alignments that is specially designed for evaluation and comparison of multiples sequence alignment programs. The database contains 6255 sequences with full length and homologous sections offering us to compare the quality for both global and local alignments. There are total 180 reference alignments. These sequences and the reference alignments are divided into five hierarchical reference sets. For each set, the sequences are constructed into the test cases so that there is a file contain the sequences and another file is the corresponding multiple alignment. In this section we will take a look at how the sequences and alignments are produced. And then we will go into the details of each reference set and its characteristics so that later on we will observe how each method will perform in which multiple alignment problems.

4.1.1 Material and Methods for generating BaliBASE

The database has been built automatically combining 3 main processes in order to generating 3 levels of alignments in which each alignment is refined and verified carefully: *Primary Structure alignments,* the *Sequence/Structure Alignments* and finally the *Construction of Reference Sets.*

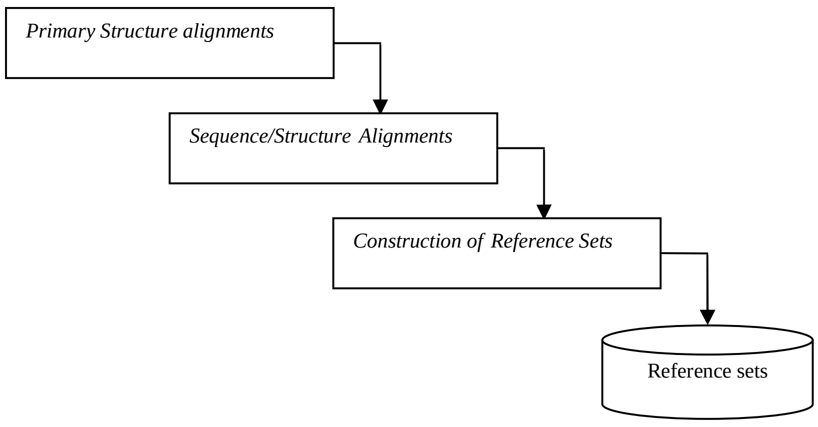
Primary Structure alignments
Sequence/Structure Alignments
Construction of Reference Sets
Reference sets

Figure 4.1.1.1:The three main process of generating BAliBASE reference

The first process *Primary Structure alignments* is the process of collecting the sample set of data and producing the first level of alignment based on the 3D structures. For each member in the original reference 1, they use PSI_BLAST to search in the PDB database with 5 iterations for each search. After that all the sequences detected with E< 10^{-3} are initially selected. In order to create challenging test cases, all the very similar sequences with >40% identity are removed. They align the sequence using MAFFT [MAFFT] program just to find the similarity of the sequences then cluster them and remove the very similar sequences. The selected sequences are then run through the SAP superposition program, automatically produce structure alignment. The structure alignment is then manually verified and refined in such a way that the alignment must contain conserved residues.

To create a larger set of test cases but still insure the quality and can be correctly verified, they noticed that there is a relatively small set of divergent sequences for each protein family from the PDB database which can be accurately aligned. In the *Sequence /Structure alignments process*, they incorporate the current selected data with Uniprot database by running BlastP to search for all the sequences with E< 10^{-3} for each of the PDB sequences in the Primary structure alignment. Similar to step1, the selected sequences then will be clustered to remove all the sequences sharing more than > 80%

residue and all the alignment that sharing < 40% similarity. The result then again is verified and refined manually. After this step, they have the complete set of PDB sequences and their related Uniprot sequences.

In the last process of *Construction of reference Sets*, they classify the sequences and the alignments into 5 Reference Sets. The reference sets are organized from the Primary multiple alignment in such a way that:

- Reference 1 contains all the equi-distant sequences with 2 different levels of conservation: the selected set of PDB with the condition that any 2 sequences share < 20% identity and no sequence has > 35% internal insertions. Among the rest: Protein families that have at least four structures are available are included in the V1 subset of the reference 1. Proteins families that have a set of at least four equidistant sequences, in which any 2 sequences share 20-40% identity are included in the new V2 subset.
- Reference 2 contains all the families aligned with a highly divergent "Orphan" sequences that share < 20% identity with all members in the family. All the sequences must belong to PDB databases since they are highly divergent enough.

- Reference 3: subgroups with <25% residue identity between groups like clustering: constructed in subfamilies so that the sequences within a given subfamily share > 40% identity but two sequences from different families share < 20% identity. Each family must contain at least one PDB sequence.

- Reference 4 sequences with N/C-terminal extensions: contains all the sequences that contain large N/C-terminal extensions that share > 20% identity with at least one other sequence.
- Reference 5: the same as ref 4 but for internal insertions.

Then, using NorMD program, they measure the sequence conservation in a sliding window analysis with the length of 5 along the alignment. By this way they define core blocks represent the regions that are reliably aligned. A core block is a region in the alignment consisting of at least three columns with no gaps with a condition that in the region either the sequences with known 3D structure all share the same secondary structure and the NorMD score is > 0.1 or the NorMD is > 0.2. Using DSSP program, they calculate the secondary structure elements for each PDB sequence. For Uniprot sequences, they use the feature table to show all the domains, signal sequences, potential transmembrane regions, binding sites etc. They also search in Pfam database for protein family domains for Uniprot sequences in each alignment. By this way they also take in to account the structure/functional information from external sources and produce the high quality test cases [BAliBASE].

4.1.2 Summary of the BAliBASE reference sets

The BAliBASE reference sets can be summarized in the table below:

Folder names	Reference	Property	Number of test cases
RV11	Reference 1	Equi-distant sequences of similar length with very divergent conservation (<20% identity)	BB11001-BB11038 are 38 cases for the full-length sequences BBS11001-BBS11038 38 cases for homologous regions only of the same sequences.
RV12	Reference 1	Equi-distant sequences of similar length with medium to divergent sequences (20-40% identity)	BB12001-BB11044 are 44 cases for the full-length sequences BBS12001-BBS12044 are 44 cases for homologous regions only.
RV20	Reference 2	Families aligned with a highly divergent "orphan" sequence	BB12001-BB11044 are 44 cases for the full-length sequences BBS12001-BBS12044 are 44 cases for

			homologous regions only.
RV30	Reference 3	Equidistant divergent families or subgroups with < 25% residue identity between groups	BB20001-BB20044 are 44 cases for the full-length sequences BBS12001-BBS12044 are 44 cases for homologous regions only.
RV40	Reference 4	Sequences with N/C-terminal extension	BB40001-BB40049 are 49 cases for the full-length sequences
RV50	Reference 5	Internal insertions	BB50001-BB50016 are 16 cases for the full-length sequences BBS50001-BBS50016 are 16 cases for homologous regions only.

Table 4.3.2: The Summary of the BAliBASE reference sets

5. Implementation

In this chapter we will discuss about how to compare the methods above and how we will design a unified experiment so we can test all the methods to see if they perform as expected.

What constitutes algorithms evaluation can be done in so many ways but in this thesis we consider 3 main requirements:

- The judgments for each method must be trustable,
- The experiment must consider different situations of multiple sequence alignment
- Problems and the results of the evaluation need to be clear and be able to trace back if needs.

Following this, we will present the experiment as well as how we construct the implementation. And finally, we will show the results of the experiment and analysis the results.

5.1 Database preparation

A good comparison for the programs requires a large number of accurate reference alignments to be used as test cases. There are several available brenchmarks for multiple sequence alignments: FREFAB [MUSCLE], OXBench, SABmark. In these methods, large data sets are provided using automatic structure superposition methods but the weak point is the automation of the alignment construction process doesn't ensure the quality of the alignments. Beside, in these brenchmarks, there is no detailed documentation of how the performance of the data was produced or test under different conditions. So it is hard to use these data sets to be test cases since we don't really know how properties of each of the sequences in the data. HOMSTRAD is an alternative method that is semiautomatic structural alignment but it is not specifically designed as a benchmark and only contains homologous protein families. In this thesis we choose BaliBASE , one of the first large scale benchmarks specifically designed for multiple sequence alighment to be the test cases. BaliBASE has several techniques to ensure the quality of the alignments. BaliBASE bases on 3D structure superpostions that are manually refined to ensure the correct aligment of conserved residues. Beside the alignments are organized into reference sets that are designed for different real multiple alignment problems. The document is well designed with detailed methods that has been running to produce the alignments as we have described in the previous chapter.

After choosing the database for the experiment, we will need to prepare a database structure of the experiment. The experiment will run through a database of test cases in which a test case is considered as below:

Definition 5.1.1*: A test case* contains the set of input sequences and the corresponding reference multiple alignments that is verified as high quality.

For each reference set in BaliBASE, all the ".TFA" files which contain non-aligned sequences in FASTA [FASTA] format will be used as input test cases:

```
>1aab_
GKGDPKKPRGKMSSYAFFVQTSREEHKKKHPDASVNFSEFSKKCSERWKT
MSAKEKGKFEDMAKADKARYEREMKTYIPPKGE
>1j46_A
MQDRVKRPMNAFIVWSRDQRRKMALENPRMRNSEISKQLGYQWKMLTEAE
KWPFFQEAQKLQAMHREKYPNYKYRPRRKAKMLPK
>1k99_A
MKKLKKHPDFPKKPLTPYFRFFMEKRAKYAKLHPEMSNLDLTKILSKKYK
ELPEKKKMKYIQDFQREKQEFERNLARFREDHPDLIQNAKK
>2lef_A
MHIKKPLNAFMLYMKEMRANVVAESTLKESAAINQILGRRWHALSREEQA
KYYELARKERQLHMQLYPGWSARDNYGKKKKRKREK
```

Example 5.1.1: An example of an input test case in FASTA format

To make it easy to mention about the three programs we are working with, we will call our chosen programs for the experiment: ClustalW, T-Coffee and MUSCLE as the test programs.

Definition 5.1.2: A test alignment is an alignment that is produced by one of the test program.

Definition 5.1.3*:* A reference alignment is the verified multiple alignment supplied by BaliBASE reference.

For each algorithm running an input test case, the test alignment and reference alignment will be given in MSF file [MSF]:

```
PileUp



   MSF:   96  Type: P    Check:  7038   ..

 Name: 1aab_ oo  Len:   96  Check:  4681  Weight:  10.0
 Name: 1j46_A oo  Len:   96  Check:  1914  Weight:  10.0
 Name: 1k99_A oo  Len:   96  Check:  8221  Weight:  10.0
 Name: 2lef_A oo  Len:   96  Check:  2222  Weight:  10.0

//



1aab_        ...GKGDPKK PRGKMSSYAF FVQTSREEHK KKHPDASVNF
SEFSKKCSER
1j46_A       ......MQDR VKRPMNAFIV WSRDQRRKMA LENP..RMRN
SEISKQLGYQ
1k99_A       MKKLKKHPDF PKKPLTPYFR FFMEKRAKYA KLHP..EMSN
LDLTKILSKK
2lef_A       ........MH IKKPLNAFML YMKEMRANVV AEST..LKES
AAINQILGRR


1aab_        WKTMSAKEKG KFEDMAKADK ARYEREMKTY IPPKGE.... ......
1j46_A       WKMLTEAEKW PFFQEAQKLQ AMHREKYPNY KYRPRRKAKM LPK...
1k99_A       YKELPEKKKM KYIQDFQREK QEFERNLARF REDHPDLIQN AKK...
2lef_A       WHALSREEQA KYYELARKER QLHMQLYPGW SARDNYGKKK KRKREK
```

Example 5.1.2:An example of a produced multiple alignment in MSF format

The order of organizing the test cases will follow the structure of BaliBASE so each file of imput sequences, produced alignment and BaliBASE verified alignment will be prefixed with the name of the test case. For instance: for the test case BB11001, the input sequences will be in file *BB11001.tfa,* after run the file through the three multiple algorithms, *BB11001_clustalw.msf, BB11001_tcoffee.msf, BB11001_muscle.msf* will be the alignment produced by clustalW, T-Coffee and MUSCLE accordingly which can then be compared with the BaliBASE verified multiple alignment *BB11001.msf* .

5.2 Alignment evaluation score

The basic idea of the experiment is for a test case, the set of input sequences will be the input for a test program to produce a test alignment which then will be compared to the BaliBASE reference alignment to give a score. There are two different scoring systems that can be used to estimate the quality of an alignment compared to a reference alignment: *The sum-of-pairs score (SPS)* and *The Column score (CS).* [A comprehensive comparison of multiple sequence alignment programs]

5.2.1 The sum-of-pairs score (SPS)

The sum-of-pairs score (SPS): is calculated so that the score increases with the number of sequences correctly aligned which is used to determine the extent to which the programs succeed in aligning. The SPS score is defined as below:

Considering a test alignment of size *NxM,* and a reference alignment of size NxMr, where *N* is the number of sequences, and *M,Mr* are the number of columns in the test and reference alignment accordingly and A_{i1} A_{i1}, ..., A_{iN}, is the *ith* column in the alignment, for each pair of residues A_{ij} and A_{ik} we define $p_{ijk} = 1$ if residues A_{ij} and A_{ik} are aligned with each other in the reference alignment, otherwise $p_{ijk} = 0$. The score S_i for the ith column will be the sum of p_{ijk} for all pairs of symbols in this column:

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

Similarly S_{ri} is the score S_i for the ith column in the reference alignment.
The SPS score for the test alignment is:

$$SPS = \sum_{i=1}^{M} S_i / \sum_{i=1}^{Mr} S_{ri}$$

5.2.2 The Column score (CS)

The Column score (CS): Considering a test alignment of size *NxM,* and a reference alignment of size NxMr, where *N* is the number of sequences, and *M,Mr* are the number of columns in the test and reference alignment accordingly: the score $C_i = 1$ if all the residues in the column are aligned in the reference alignment, otherwise $C_i = 0$-

The CS score for the test alignment is then:

$$CS = \sum_{i=1}^{M} C_i / M$$

Since the two scoring systems have been implemented successfully in the program BaliBASE called *Bali_score* which takes as input a test alignment and a reference alignment in MSF format, in this thesis we will use the Bali_score to estimate the quality of a test alignment in our experiment.

5.3 Experiment implementation

We will implement two experiments: one to compare the three programs in multiple alignment qualities and the other to compare the running time. The experiment will be performed in python [python] and the results will be analyzed using Microsoft excel.

5.3.1 The quality experiment

In order to implement the quality experiment, we divide the big implementation into 3 main steps:

Step 1: Producing multiple alignments

For each reference in the BaliBASE, we will consider 2 groups of sequences, the homologous region sequences and the full-length sequences. The homologous region sequences are named with prefix BBS and all the full-length sequences are named as BB, for instance BB11001.tfa contain full-length sequences and BBS11001.tfa contains the homologous regions of the same set of sequences. Each test program takes a .tfa file as the input and produce the test multiple alignment in MSF format.

The producing multiple alignment step can be in the following pseudo-codes

```
Step 1: Producing multiple alignments

For each reference in the BaliBASE do:
    For each groups do:
        For test case do:
            For each test program do:
                ▪ Run the multiple alignment program
                ▪ Output the test alignment
```

Step 2: Calculate the BaliBASE scores:

```
Step 1: Producing multiple alignments

For each produced test alignment:
                ▪ Run Bali_score on the test alignment and
                  the corresponding reference alignment
                ▪ Output the two scores (SPS and CS)
```

In order to make it easier to keep track about the input and output for each test program and for the Bali_score program. The implementation actually combines step 1 and 2 into one procedure as below:

```
Step 1: Producing multiple alignments

For each reference in the BaliBASE do:
    For each groups do:
        For test case do:
            For each test program do:
                ▪ Run the multiple alignment program
                ▪ Output the test alignment
                ▪ Run Bali_score on the test alignment and the
                  corresponding reference alignment
                ▪ Output the two scores (SPS and CS)
```

Step 3: Collecting and analyzing the results

We will build the comparison for 4 different categories for each reference.

- Comparing the CS scores of 4 methods on the full length sequences
- Comparing the SP scores of 4 methods on the full length sequences
- Comparing the CS scores of 4 methods on the homologous regions only.
- Comparing the SP scores of 4 methods on the homologous regions only.

The scores will be collected into the 4 comparing categories and presented as graphs. In this step, we will use Microsoft excel as the tool for analyzing and drawing graphs.

5.3.2 The running time experiment

The idea of the running time experiment can be demonstrated in the pseudo-codes below:

Step 1: Producing multiple alignments

```
For random reference in the BaliBASE do:
        For test case do:
            For each test program do:
                ▪ Measure the running time of
                    • Run the multiple alignment program
                    • Output the test alignment
```

We will choose some random reference in BaliBASE and measure the running time that each test program uses to produce the multiple sequence for each test case. We use the standard command *time* of Linux to measure the running time of a test program, i.e. the elapsed real time between invocation and termination in seconds. The running time for all the test cases in a reference will be collected, compared and presented as graphs.

6. Evaluation

In this chapter we will evaluate the results of the implementations we have described in the previous chapter. There are two main implementations: the quality experiment and the running time experiment.

6.1 The quality experiment

6.1.1 Quality comparison for the reference 1-RV11 of BaliBASE

(Note: in the graphs, the CS score is more strictly. To avoid to have a bias in favour of BaliBASE reference, we will consider the SPS scores mainly and use CS score as reference to confirm our comments)

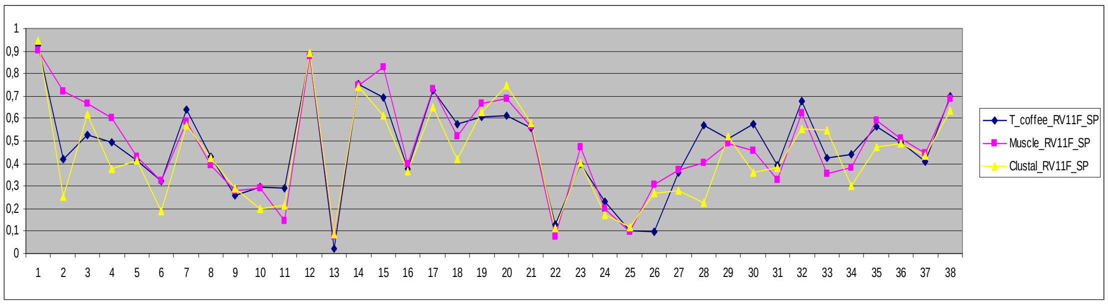

Figure 6.1.1.a: SPS scores quality comparison with full-length sequences for the reference 1-RV11

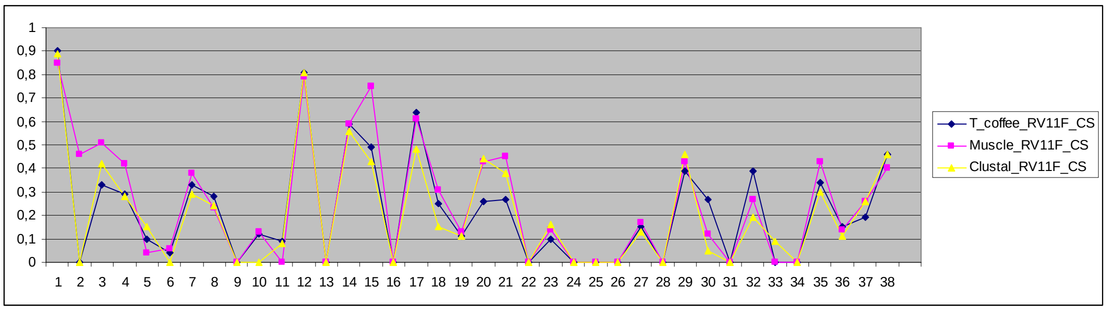

Figure 6.1.1.b: CS scores quality comparison with full-length sequences for the reference 1-RV11

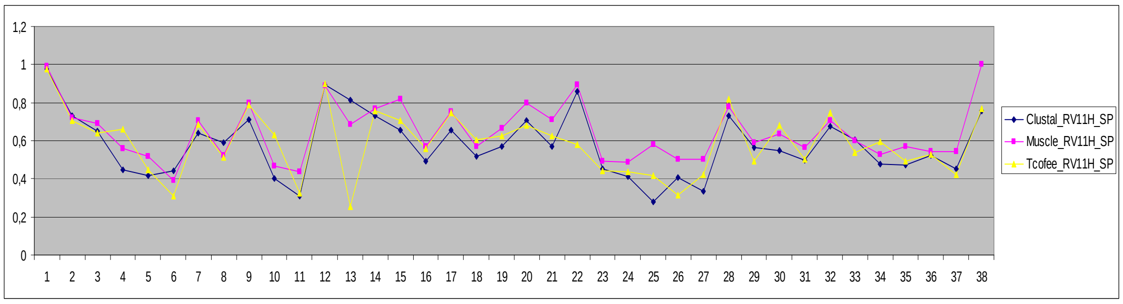

Figure 6.1.1.c: Quality comparison with SP scores with homologous regions for the reference 1-RV11

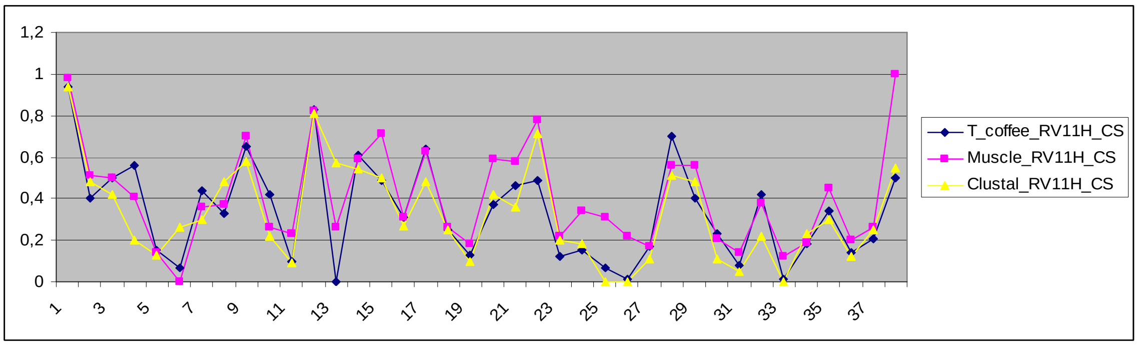

Figure 6.1.1.d: CS scores quality comparison with homologous regions for the reference 1-RV11

Figure 6.1.1.a - d show the results of the quality experiment on the first reference RV11 of BaliBASE. The graph are created from the database of the BaliBASE score results in which each program running on each test case in the reference [Appendix B]. The vertical line of each graph indicates the BaliBASE scores which value from 0 to 1. The higher value of the BaliBASE score indicates the better the quality of the test alignment. The horizon line of each graph show the ordinal number of each case in the reference and there are 38 test cases for full-length sequences and 38 test cases for homologous regions in the RV11 reference. Each line of a graph represents the baliBASE scores of a test program. The score is running from *(0,1)* and the higher the value of the score means the higher quality of the test alignment. Each line is named with the notation of the method: the reference set with either full length sequences (F) or for the homologous regions (H) and the score that it presents. For example if the line named as:

Tcoffee_RV11F_SP means this line presents the SPS score performance of Tcoffee with the Reference 1 data set named RV11 with full length sequences

Recall that RV11 contains Equi-distant sequences of similar length with very divergent conservation (<20% identity). In the four graphs all the three programs almost have the same patterns and chasing each other in the quality scores. There are so many points that the three programs all have the high scores in the same test cases and low scores in other test cases. The pink line (Muscle) shows a stable quality of alignment and higher than the other programs. The T-coffee and Clustalw shows almost the same quality in average but for some cases for instance 28 and 13 for homologous data cases they have quite different in scores. The program seems process better for homologous cases. But in general they follow the same patterns which mean the quality of the alignments producing by the 3 programs is stable and not much difference except for the case 13.

6.1.2 Reference 1-RV12

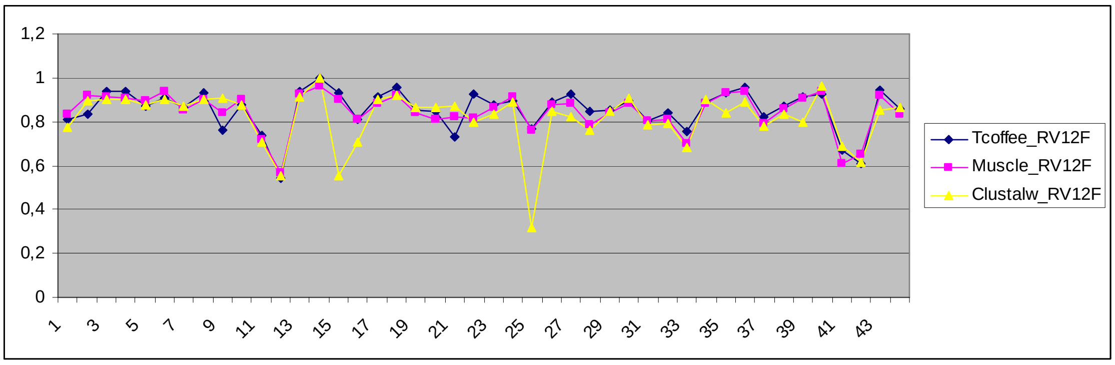

Figure 6.1.2.a: SPS scores quality comparison with full-length sequences for the reference 1-RV12

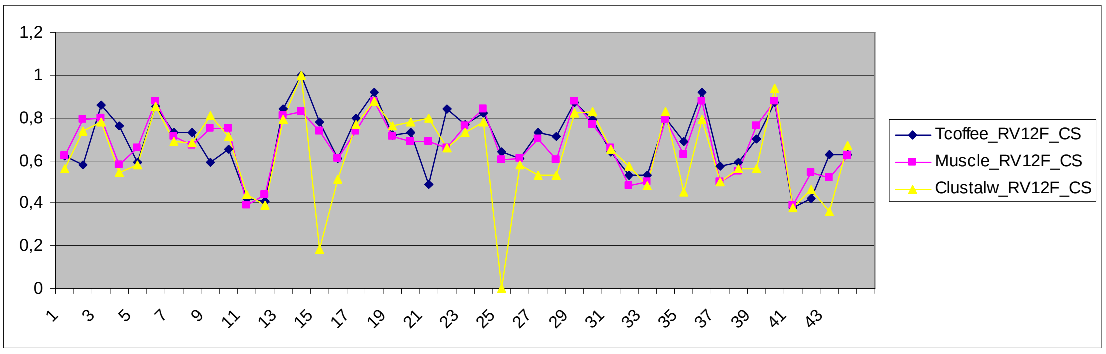

Figure 6.1.2.b: CS scores quality comparison with full-length sequences for the reference 1-RV12

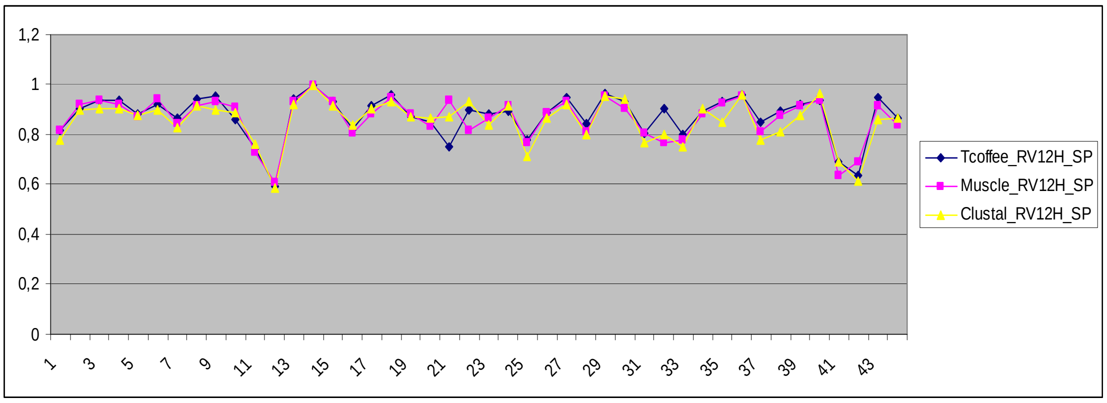

Figure 6.1.2.c: Quality comparison with SP scores with homologous regions for the reference 1-RV12

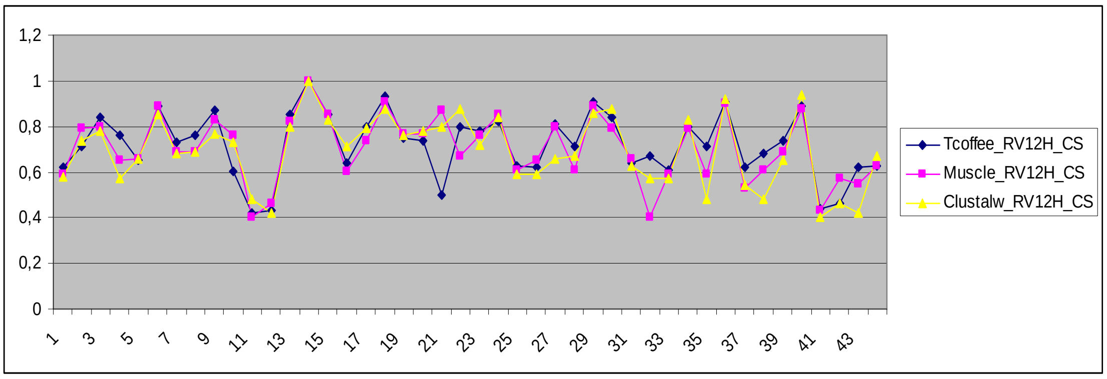

Figure 6.1.2.d: CS scores quality comparison with homologous regions for the reference 1-RV12

Figure 6.1.2.a - d show the results of the quality experiment on the Reference 1-RV12 of BaliBASE which contains equi-distant sequences of similar length with medium to divergent sequences (20-40% identity). Similar to the performance of the test programs on the reference 1-RV11, the patterns of the lines are similar between the 3 lines specially for the homologous sequences. In this reference set, T-coffee some how shows a little better scores than other programs. Muscle and ClustalW show almost the same scores as each other except for some special cases such as cases 26 and 16, then Clustalw drop very low compare to other programs. In the reference RV12, T-Coffee and MUSCLE both produce better scores than ClustalW but we can not say which of them are better in these graphs.

6.1.3 Reference 2 – RV20

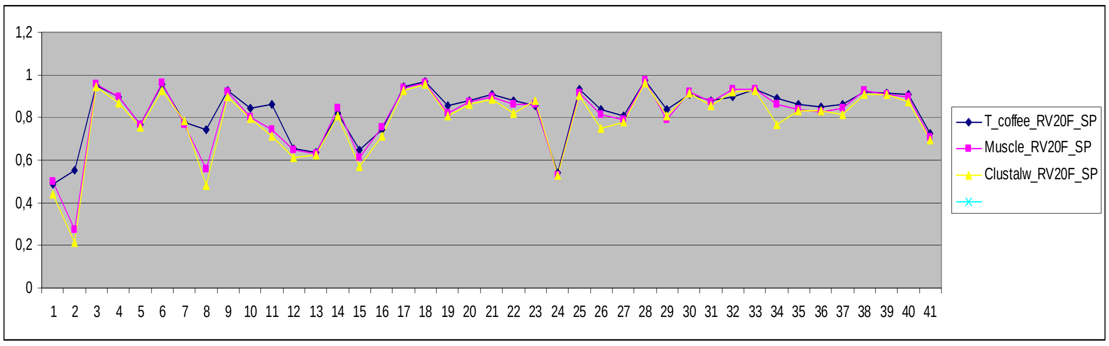

Figure 6.1.3.a: SPS scores quality comparison with full-length sequences for the reference 2-RV20

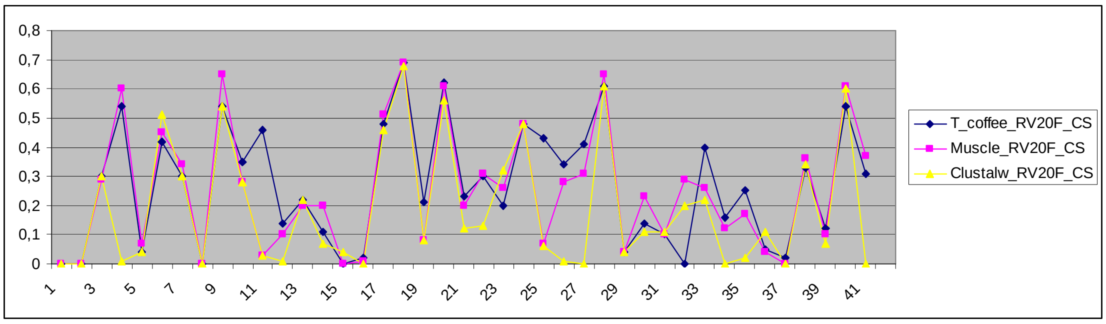

Figure 6.1.3.b: CS scores quality comparison with full-length sequences for the reference 2-RV20

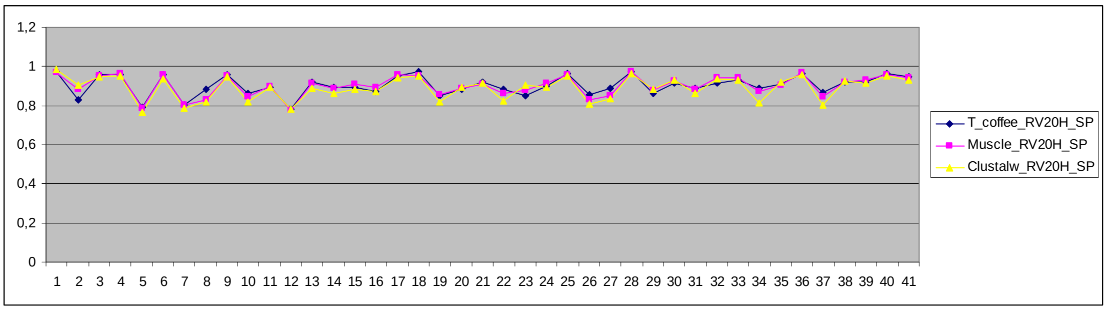

Figure 6.1.3.c: Quality comparison with SP scores with homologous regions for the reference 2-RV20

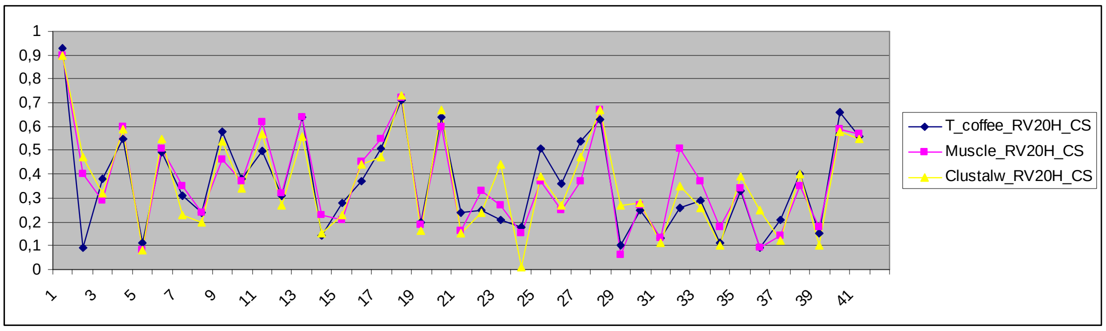

Figure 6.1.3.d: CS scores quality comparison with homologous regions for the reference 2-RV20

Figure 6.1.3.a - d show the results of the quality experiment on the Reference 2-RV20 of BaliBASE which contains families aligned with a highly divergent "orphan" sequence. We can also see the three lines also have the same patterns in SPS scores but they are pretty much different in CS scores. It shows that for the same test cases the three programs have produced three different solutions for multiple alignments. The SPS scores are competitive and go up and down almost at the same points, which show the three programs are stable and competitive. In these graphs, the blue line always slightly higher than other lines which means T-Coffee give the best quality scores in general, after T-Coffee it is MUSCLE and then ClustalW.

6.1.4 Reference 3 – RV30

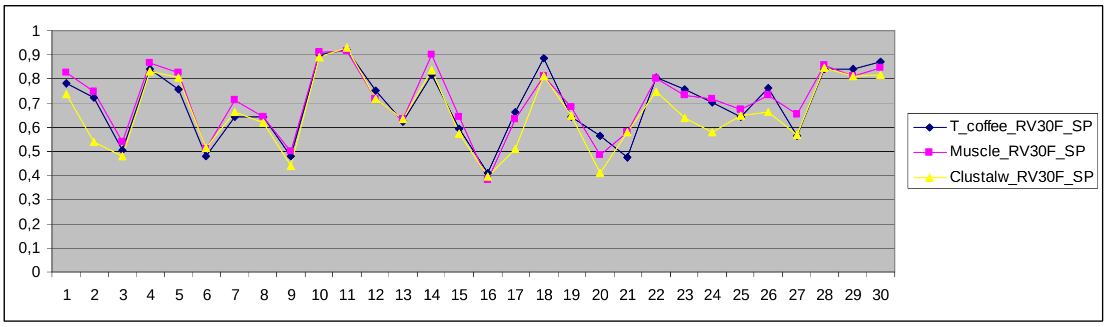

Figure 6.1.4.a: SPS scores quality comparison with full-length sequences for the reference 3-RV30

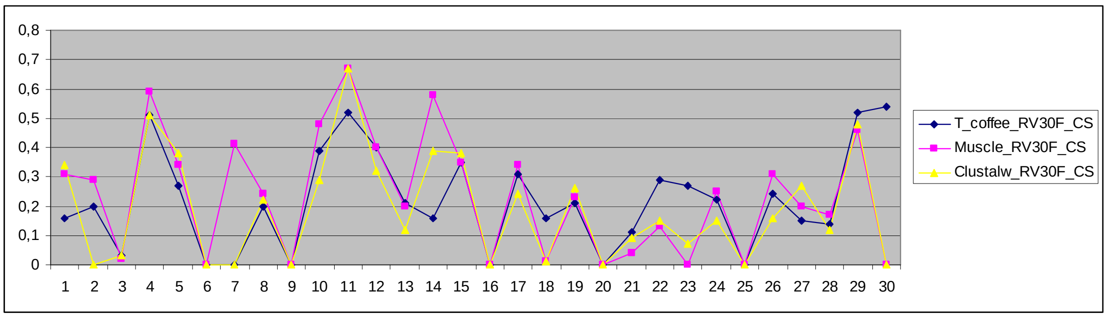

Figure 6.1.4.b: CS scores quality comparison with full-length sequences for the reference 3-RV30

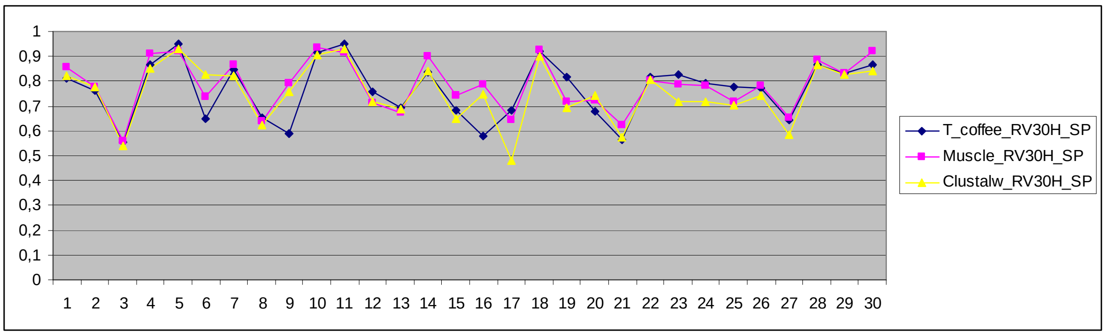

Figure 6.1.4.c: Quality comparison with SP scores with homologous regions for the reference 3-RV30

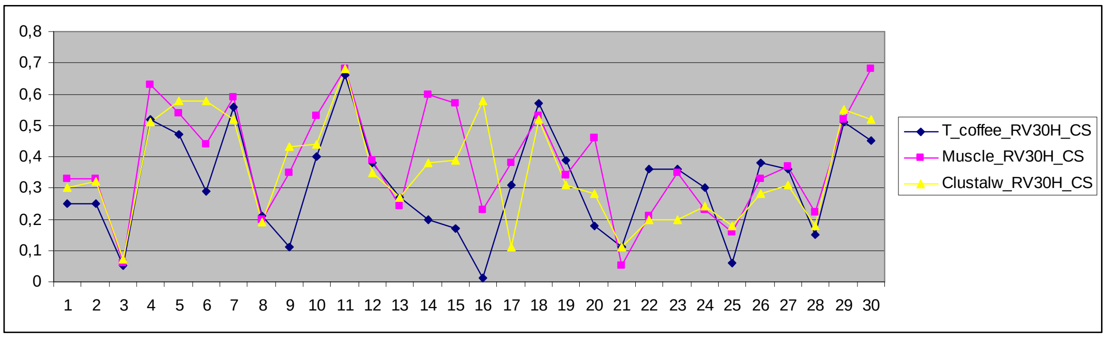

Figure 6.1.4.d: CS scores quality comparison with homologous regions for the reference 3-RV30

Figure 6.1.4.a - d show the results of the quality experiment on the Reference 3-RV30 of BaliBASE which contains divergent families or subgroups with < 25% residue identity between groups. In this experiment we can see the different patterns of test programs when processing on full-length sequences and on homologous regions. Considering the SPS scores, if figure 6.1.4.a shows that MUSCLE and T-Coffee produce as high quality as each other and better than ClustalW then figure 6.1.4.b shows that when it comes to homologous regions, then MUSCLE produces better scores than T-Coffee and ClustalW actually not much lower then T-Coffee. But it is every interesting to consider the CS scores in figure 6.1.4.b and figure 6.1.4.b which shows that ClustalW actually have better scores and respect the reference more than both MUSCLE and T-Coffee in both full-length or homologous sequences while the SPS scores in other graphs are still very competitive with other programs. We can say in this reference, ClustalW and MUSCLE produce better scores than T-Coffee.

6.1.5 Reference 4 – RV40

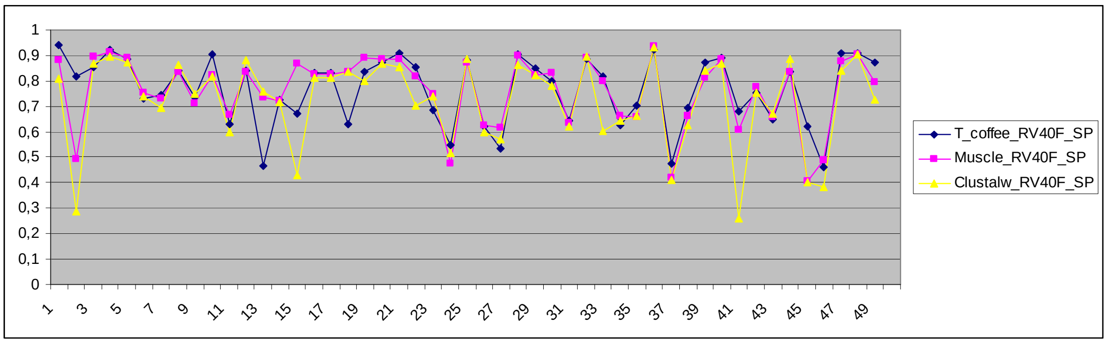

Figure 6.1.5.a: SPS scores quality comparison with full-length sequences for the reference 4-RV40

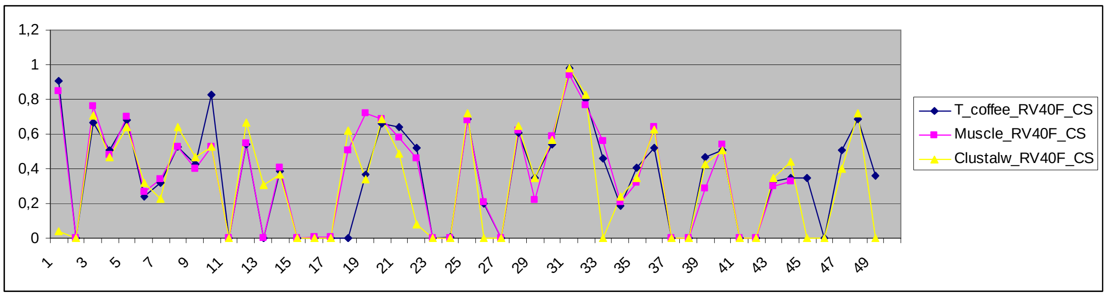

Figure 6.1.5.b: CS scores quality comparison with full-length sequences for the reference 4-RV40

Figure 6.1.4.a and b show the results of the quality experiment on the Reference 4-RV40 which contains sequences with N/C-terminal extension. MUSCLE shows higher quality over the test cases. T-coffee produces some best scores but line are not stable, they are very high in some sequences but low in others. ClustalW produces lower scores in general in this reference.

6.1.6 Reference 5 – RV50

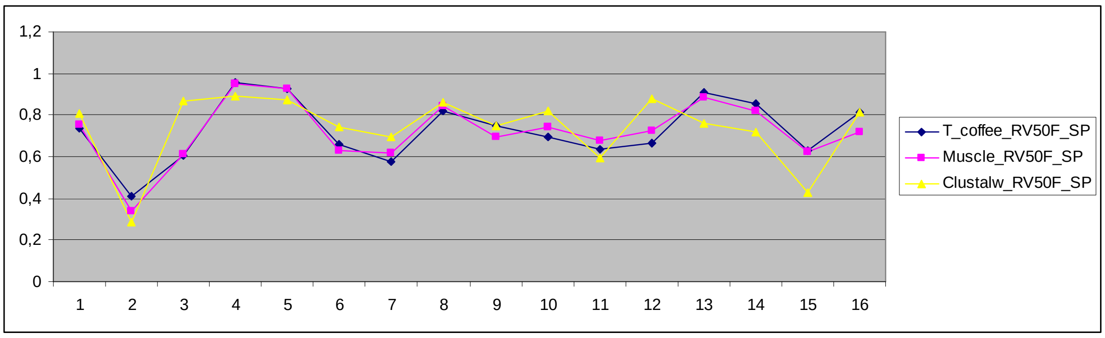

Figure 6.1.6.a: SPS scores quality comparison with full-length sequences for the reference 5-RV50

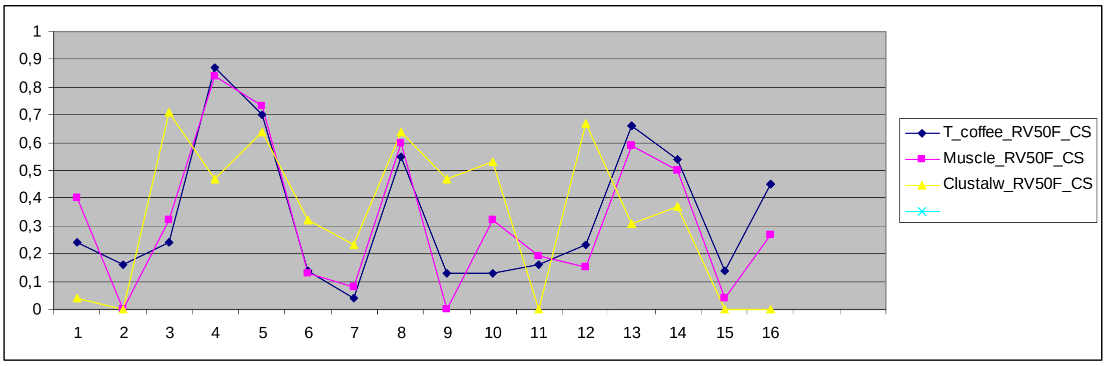

Figure 6.1.6.b: CS scores quality comparison with full-length sequences for the reference 5-RV50

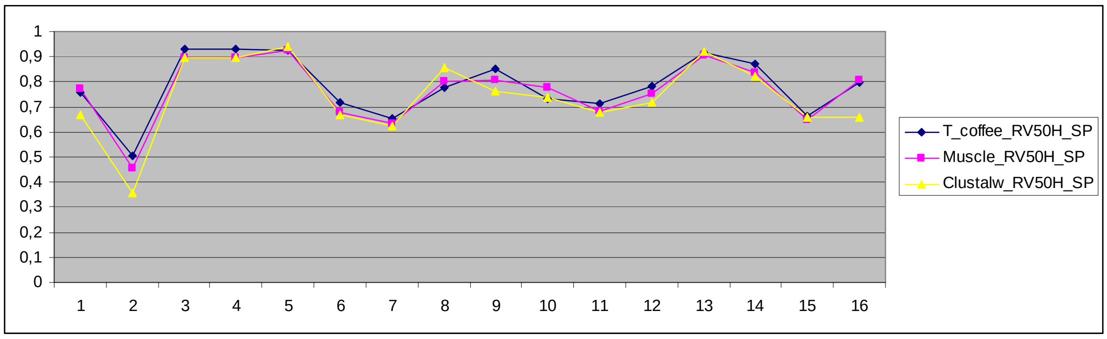

Figure 6.1.6.c: Quality comparison with SP scores with homologous regions for the reference 5-RV50

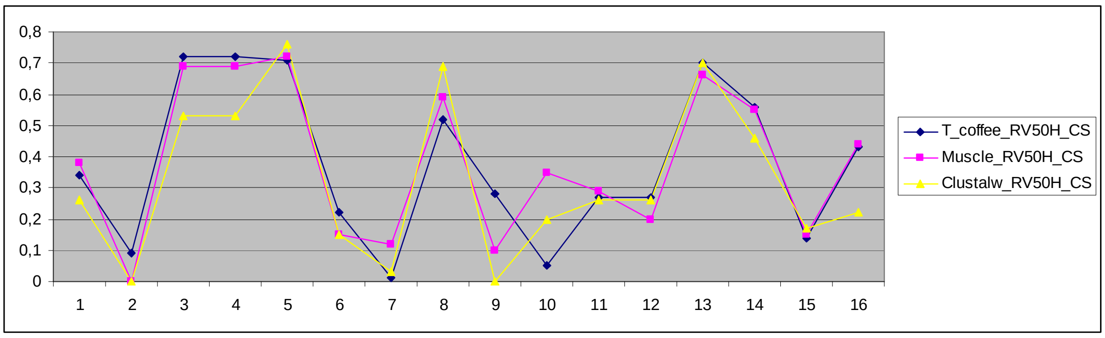

Figure 6.1.6.d: CS scores quality comparison with homologous regions for the reference 5-RV50

Figure 6.1.6.a - d show the results of the quality experiment on the Reference 5-RV50 of BaliBASE which contains with N/C- internal insertions (the sequences that contain large N/C- internal insertions that share > 20% identity with at least one other sequence).

Different from other references, Figure 6.1.6.a shows that ClustalW produces better or equal alignment SPS scores than both MUSCLE and T-Coffee for a haft of the test cases. Figure 6.1.6.b also confirms that ClustalW produces alignments with high respect to the reference: i.e for the cases 3, 9, 10, 12 clustalW produces alignments that similar to the reference alignments while MUSCLE and T-Coffee do not. the performances of the three programs for homologous regions are more in the same pattern than for the full-length sequences. MUSCLE and T-coffee are the programs who can reach the best scores regarding SPS scores and full-length sequence. For homologous regions, ClustalW reaches a best CS score on the test case 5.

6.2 The running time experiment

While processing the quality experiment, it is easy to noticed that T-coffee produces alignments in much more longer time than the two others. In some cases, clustalW can take 30 minutes to finish a reference while T-Coffee takes several hours or a haft of day to run. In this experiment we will demonstrate the difference between the running times of the three programs

Noticed from the quality experiment that the reference RV11 takes least time to process, we will choose to measure the running time of the three programs on this reference for full length sequences. The running time results are presented in the figure 6.2.1 which shows the clear differences between the three programs. The graph is created from the database of the BaliBASE running time results [Appendix B]. The vertical line of each graph indicates the running time in seconds. The horizon line of the graph shows the ordinal number of each case in the reference. The figure 6.2.1 shows that ClustalW and MUSCLE takes almost no time to process, but T-Coffee can takes much more time to run in some cases such as 5 and 18.

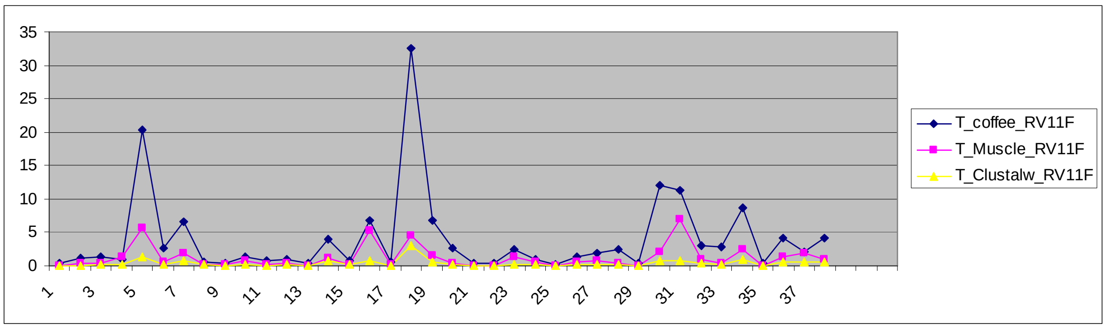

Figure 6.2.1: Running time comparison for the reference 1-RV11 for full length sequences

The running time difference between the three programs is much clearer when we perform the running time experiment on the reference 2- RV20 which results on the table 6.2.1. The values in the table is in minute (m) and second. It is hard to view the comparison on graphs because the difference on the running time scale in which some of the test cases take seconds while the others can take up to an half an hour to run. In the Table 6.2.1 we can see clearly that T-Coffee is extremely slower compare to the two others. There are so many cases that take only seconds on MUSCLE and ClustalW to run but it can take up to 15 or 30 minutes on T-Coffee. Like we expected, MUSCLE can produces alignments with very much the same speed as ClustalW. For some cases MUSCLE is faster and for other cases ClustalW is faster but the difference is just in seconds.

Nr.	T_coffee_RV20F	T_Muscle_RV20F	T_Clustalw_RV20F
1	21.791	4.007	1.466
2	1m51.489	41.223	6.708
3	24.304	13.709	45.562
4	13m34.552	16.281	51.179
5	4m12.503	5.541	12.476
6	4m32.333	2.753	7.708
7	1m4.731	2.703	3.945
8	14m13.840	1m17.752	27.599
9	1m28.929	2.640	5.005
10	6m6.466	19.440	40.173
11	22.210	2.527	0.711
12	1m33.325	6.225	5.708
13	2m19.991	6.670	8.609

14	9m0.591	14.618	13.738
15	5m6.675	2.542	2.390
16	42.510	3.362	1.297
17	8m36.101	9.511	37.646
18	6m22.519	4.617	13.198
19	1m33.214	3.268	6.262
20	19.043	1.122	1.152
21	14m5.810	16.980	54.919
22	8m34.916	6.463	17.050
23	54.923	1.288	2.002
24	2m51.127	10.412	5.887
25	26m27.984	14.211	53.129
26	3m35.621	18.809	16.564
27	2m57.836	10.525	7.642
28	5m27.987	7.250	13.148
29	26.148	0.576	0.417
30	1m45.504	0.970	1.929
31	5m31.193	5.308	6.781
32	3m41.521	1.955	2.161
33	1m43.956	1.080	1.270
34	7m31.937	9.317	10.926
35	4m44.084	26.686	23.909
36	15m13.476	7.879	5.174
37	16m26.980	15.598	41.544
38	1m22.051	1.152	1.087
39	33m19.270	15.358	1m2.187
40	31m29.051	20.873	1m16.699
41	14m16.976	48.914	37.733

Table 6.2.1: Running time comparison for the reference 2-RV20 for full length sequences

This running time experiment clearly shows that MUSCLE and ClustalW are fast multiple sequence alignment programs that can process with a big number of sequences. T-Coffee is only fast for sequences with short lengths, but when it comes to complex and long sequences then T-Coffee will take a lot of time to run.

7. Conclusion and Future work

The main goal of this thesis is to implement an experiment on different multiple sequence alignment programs on practice so that we can review back their methodologies as well as their statements about their advantages. The thesis intends to compare these algorithms by creating a unified experiment environment in which every program will use the same input and output and then compare the results on both the accuracy and running time aspects. In order to insure the quality of the comparison, this thesis chooses to run a full experiment on the BaliBASE reference so we can the quality alignment of each test case. All the programs are processed with default parameters with is the most common used by normal users.

The quality experiment shows that the three programs are stable and completive in quality. MUSCLE and T-Coffee are normally can reach the best quality scores according to the BaliBASE reference and Bali_score estimation and show higher scores in general than ClustalW. In general ClustalW can produce good scores but in some special cases the scores can suddenly dropped compare to two other programs. The three programs also show that they can face the multiple alignment problems as good as each other by giving the same patterns in the graphs. The conclusion for the quality measures for the three programs is: they are very much competitive in quality and it is hard to tell the differences between the three programs in general. But as we expected, MUSCLE and T-Coffee are considered as high quality multiple sequence alignment programs.

Different from the quality experiment, the three programs show difference in running time. MUSCLE and ClustalW can surprise user by the capability of handling big number of sequences and complex sequences with a small running time. T-Coffee is much slower in running time specially for big number of sequences or divergent sequences.

Overall the thesis has proved that the programs processed as their papers have state but there are more experiments that can be implemented in future. Including in the codes listing [Appendix], there is a small test about the reverse alignment i.e. the reversed alignment of all reverse sequences should be the same as the alignment of non-reversed sequences that can be tested on a single test case. This reverse alignment experiment should be an interesting experiment on high- throughput data.

In the beginning this thesis also wanted to compare MAFFT program [MAFFT] to other multiple alignment programs but there is a difficulty of converting the output of multiple alignment in clustalW format to MSF format while calculating the Bali_scores. Another further implementation could be done is making a input and output converter for the experiment so that different multiple alignment formats to the MSF format and all the other sequences files will be converted to TFA files, so that when we need to compare another multiple programming it can be added to the experiment easily.

Reference

[BAliBASE]: A benchmark alignments database for the evaluation of multiple sequence alignment programs. Julie Thompson et al.

[HollyGrail]: Algorithms on Strings, Trees and Sequences. Gusfield, Dan - chapter 14

[Waterman]: Waterman MS, Smith TF, Beyer WA: Some biological sequence metrics. Adv in Math 1976, 20:367-387.

[Edgar]: Edgar, R.C.(2004) MUSCLE: a multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics 5(1):113.

[Goldberg]: Goldberg, David E (1989) Genetic Algorithm in Search, Optimization and Machine learning, Kluwer Academic Publishers, Boston, MA].

[T-Coffee]. T-Coffee: A Novel Method for Fast and Accurate Multiple Sequence Alignment. Cedric Notredame et al.

[MUSCLE]: MUSCLE: a multiple sequence alignment method with reduced time and space complexity.Robert C Edgar.

[Clustal W]: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Julie D.Thompson et al.

[MAFFT]: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. Katoh K et al.

[FASTA]: http://en.wikipedia.org/wiki/Fasta_format

[MSF]: http://www.bioperl.org/wiki/MSF_multiple_alignment_format

Appendix

A. Code listing

All the executive files are in the folder /users/u050004/mythesis/muscle3.6/

The results will be extract to folder: /users/u050004/mythesis/

The quality experiment

P1 indicates the full- length sequences and p2 indicates homologous regions

File names	Notes
run_bali_RV11_clustalw_p1.py run_bali_RV11_muscle_p1.py time_bali_tcoffe_RV11_p1.py	Quality experiment on RV11-full length
run_bali_RV11_clustalw_p2.py run_bali_RV11_muscle_p2.py time_bali_tcoffe_RV11_p2.py	Quality experiment on RV11-homologous regions
run_bali_RV12_clustalw_p1.py run_bali_RV12_muscle_p1.py time_bali_tcoffe_RV12_p1.py	Quality experiment on RV12-full length
run_bali_RV12_clustalw_p2.py run_bali_RV12_muscle_p2.py time_bali_tcoffe_RV12_p2.py	Quality experiment on RV12-homologous regions
run_bali_RV20_clustalw_p1.py run_bali_RV20_muscle_p1.py time_bali_tcoffe_RV20_p1.py	Quality experiment on RV20-full length
run_bali_RV20_clustalw_p2.py	Quality experiment on RV20-

run_bali_RV20_muscle_p2.py time_bali_tcoffe_RV20_p2.py	homologous regions
run_bali_RV30_clustalw_p1.py run_bali_RV30_muscle_p1.py time_bali_tcoffe_RV30_p1.py	Quality experiment on RV30-full length
run_bali_RV30_clustalw_p2.py run_bali_RV30_muscle_p2.py time_bali_tcoffe_RV30_p2.py	Quality experiment on RV30-homologous regions
run_bali_RV40_clustalw_p1.py run_bali_RV40_muscle_p1.py time_bali_tcoffe_RV40_p1.py	Quality experiment on RV40-full length
run_bali_RV50_clustalw_p1.py run_bali_RV50_muscle_p1.py time_bali_tcoffe_RV50_p1.py	Quality experiment on RV50-full length
run_bali_RV50_clustalw_p2.py run_bali_RV50_muscle_p2.py time_bali_tcoffe_RV50_p2.py	Quality experiment on RV50-homologous regions
Reverse.py	To reverse a tfa. sequences file
Test_reverse_msf	To erverse a msf alignment

The running time experiment

File names	Notes
time_bali_RV11_clustalw_p1.py time_bali_RV11_muscle_p1.py time_bali_tcoffe_RV11_p1.py	Running time experiment on RV11-full length
time_bali_RV20_clustalw_p1.py time_bali_RV20_muscle_p1.py time_bali_tcoffe_RV20_p1.py	Running time experiment on RV20-full length sequences

B. Quality experiment results

Here are the Bali_scores database corresponding to the figures in chapter 6.

Figure 6.1.1.a:

	T_coffee_RV11F_SP	Muscle_RV11F_SP	Clustal_RV11F_SP
1	0,923	0,904	0,947
2	0,42	0,721	0,255
3	0,528	0,665	0,618
4	0,496	0,6	0,375
5	0,415	0,43	0,415
6	0,322	0,325	0,189
7	0,639	0,588	0,571
8	0,43	0,393	0,426
9	0,259	0,28	0,293
10	0,298	0,29	0,201
11	0,293	0,145	0,214
12	0,884	0,877	0,893
13	0,022	0,076	0,084
14	0,752	0,749	0,744
15	0,691	0,827	0,612
16	0,376	0,396	0,367
17	0,727	0,732	0,651
18	0,573	0,521	0,418
19	0,606	0,664	0,63
20	0,614	0,688	0,748
21	0,558	0,562	0,581
22	0,129	0,073	0,114
23	0,398	0,473	0,403
24	0,232	0,197	0,173
25	0,101	0,095	0,118
26	0,095	0,305	0,268
27	0,359	0,371	0,278
28	0,572	0,402	0,228
29	0,51	0,49	0,523
30	0,574	0,456	0,361
31	0,393	0,329	0,38
32	0,68	0,625	0,555
33	0,425	0,354	0,546
34	0,441	0,38	0,299
35	0,564	0,592	0,473
36	0,495	0,511	0,49
37	0,407	0,445	0,438
38	0,701	0,688	0,635

Figure 6.1.1.b:

	T_coffee_RV11F_CS	Muscle_RV11F_CS	Clustal_RV11F_CS

1	0,9	0,85	0,89
2	0	0,46	0
3	0,33	0,51	0,42
4	0,29	0,42	0,28
5	0,1	0,04	0,15
6	0,04	0,06	0
7	0,33	0,38	0,29
8	0,28	0,23	0,24
9	0	0	0
10	0,12	0,13	0
11	0,09	0	0,08
12	0,81	0,79	0,81
13	0	0	0
14	0,59	0,59	0,56
15	0,49	0,75	0,43
16	0	0	0
17	0,64	0,61	0,48
18	0,25	0,31	0,15
19	0,11	0,13	0,11
20	0,26	0,43	0,44
21	0,27	0,45	0,38
22	0	0	0
23	0,1	0,14	0,16
24	0	0	0
25	0	0	0
26	0	0	0
27	0,15	0,17	0,13
28	0	0	0
29	0,39	0,43	0,46
30	0,27	0,12	0,05
31	0	0	0
32	0,39	0,27	0,19
33	0	0	0,09
34	0	0	0
35	0,34	0,43	0,3
36	0,15	0,14	0,11
37	0,19	0,26	0,26
38	0,46	0,4	0,46

Figure 6.1.1.c:

	Tcofee_RV11H_SP	Muscle_RV11H_SP	Clustal_RV11H_SP
1	0,974	0,993	0,974
2	0,709	0,72	0,733

3	0,642	0,69	0,649
4	0,663	0,561	0,449
5	0,445	0,519	0,416
6	0,311	0,39	0,443
7	0,687	0,708	0,642
8	0,513	0,524	0,589
9	0,786	0,799	0,712
10	0,631	0,466	0,403
11	0,327	0,439	0,311
12	0,9	0,889	0,895
13	0,255	0,686	0,816
14	0,757	0,766	0,73
15	0,705	0,817	0,655
16	0,561	0,568	0,493
17	0,748	0,753	0,654
18	0,604	0,571	0,517
19	0,627	0,664	0,569
20	0,679	0,796	0,705
21	0,623	0,712	0,569
22	0,579	0,895	0,86
23	0,443	0,491	0,454
24	0,436	0,49	0,413
25	0,415	0,582	0,281
26	0,314	0,505	0,408
27	0,42	0,501	0,336
28	0,82	0,776	0,733
29	0,495	0,588	0,563
30	0,681	0,637	0,55
31	0,501	0,565	0,498
32	0,749	0,709	0,678
33	0,539	0,598	0,603
34	0,597	0,53	0,476
35	0,495	0,567	0,473
36	0,529	0,545	0,525
37	0,421	0,545	0,45
38	0,768	1	0,756

Figure 6.1.1.d:

	T_coffee_RV11H_CS	T_Muscle_RV11H_CS	T_Clustalw_RV11H_CS

1	0.940	0.980	0.940
2	0.400	0.510	0.480
3	0.500	0.500	0.420
4	0.560	0.410	0.200
5	0.150	0.140	0.130
6	0.070	0.000	0.260
7	0.440	0.360	0.300
8	0.330	0.370	0.480
9	0.650	0.700	0.580
10	0.420	0.260	0.220
11	0.100	0.230	0.090
12	0.830	0.820	0.810
13	0.000	0.260	0.570
14	0.610	0.590	0.540
15	0.490	0.710	0.500
16	0.310	0.310	0.270
17	0.640	0.630	0.480
18	0.250	0.260	0.250
19	0.130	0.180	0.100
20	0.370	0.590	0.420
21	0.460	0.580	0.360
22	0.490	0.780	0.710
23	0.120	0.220	0.200
24	0.150	0.340	0.180
25	0.070	0.310	0.000
26	0.010	0.220	0.000
27	0.170	0.170	0.110
28	0.700	0.560	0.510
29	0.400	0.560	0.480
30	0.230	0.210	0.110
31	0.080	0.140	0.050
32	0.420	0.380	0.220
33	0.010	0.120	0.000
34	0.180	0.190	0.230
35	0.340	0.450	0.300
36	0.140	0.200	0.120
37	0.210	0.260	0.250
38	0.500	1.000	0.550

Figure 6.1.2.a:

	Tcoffee_RV12F	Muscle_RV12F	Clustalw_RV12F
1	0,808	0,833	0,776
2	0,832	0,922	0,898

3	0,936	0,913	0,902
4	0,94	0,909	0,901
5	0,87	0,894	0,877
6	0,907	0,938	0,901
7	0,864	0,853	0,871
8	0,93	0,903	0,901
9	0,759	0,842	0,905
10	0,875	0,9	0,879
11	0,739	0,718	0,708
12	0,544	0,567	0,553
13	0,937	0,923	0,911
14	1	0,964	1
15	0,931	0,904	0,552
16	0,808	0,808	0,708
17	0,915	0,883	0,899
18	0,955	0,921	0,918
19	0,852	0,841	0,865
20	0,844	0,813	0,867
21	0,732	0,823	0,871
22	0,927	0,819	0,796
23	0,88	0,865	0,834
24	0,894	0,913	0,888
25	0,768	0,763	0,315
26	0,889	0,875	0,846
27	0,925	0,885	0,825
28	0,848	0,788	0,76
29	0,855	0,84	0,848
30	0,895	0,885	0,908
31	0,807	0,802	0,785
32	0,841	0,813	0,789
33	0,754	0,701	0,681
34	0,892	0,884	0,903
35	0,929	0,93	0,842
36	0,959	0,94	0,888
37	0,821	0,791	0,777
38	0,87	0,856	0,835
39	0,915	0,91	0,795
40	0,926	0,937	0,963
41	0,672	0,609	0,687
42	0,609	0,651	0,615
43	0,943	0,921	0,851
44	0,857	0,835	0,863

Figure 6.1.2.b:

	Tcoffee_RV12F_CS	Muscle_RV12F_CS	Clustalw_RV12F_CS

1	0,62	0,62	0,56
2	0,58	0,79	0,74
3	0,86	0,8	0,78
4	0,76	0,58	0,54
5	0,59	0,66	0,58
6	0,85	0,88	0,85
7	0,73	0,71	0,69
8	0,73	0,67	0,68
9	0,59	0,75	0,81
10	0,65	0,75	0,71
11	0,42	0,39	0,44
12	0,41	0,44	0,39
13	0,84	0,81	0,79
14	1	0,83	1
15	0,78	0,74	0,18
16	0,61	0,61	0,51
17	0,8	0,74	0,77
18	0,92	0,88	0,88
19	0,72	0,71	0,76
20	0,73	0,69	0,78
21	0,49	0,69	0,8
22	0,84	0,66	0,66
23	0,77	0,76	0,73
24	0,82	0,84	0,78
25	0,64	0,6	0
26	0,61	0,61	0,58
27	0,73	0,7	0,53
28	0,71	0,6	0,53
29	0,87	0,88	0,82
30	0,79	0,77	0,83
31	0,64	0,66	0,65
32	0,53	0,48	0,57
33	0,53	0,5	0,48
34	0,8	0,79	0,83
35	0,69	0,63	0,45
36	0,92	0,88	0,79
37	0,57	0,5	0,5
38	0,59	0,55	0,56
39	0,7	0,76	0,56
40	0,87	0,88	0,94
41	0,38	0,39	0,38
42	0,42	0,54	0,46
43	0,63	0,52	0,36
44	0,63	0,62	0,67

Figure 6.1.2.c

Tcoffee_RV12H_SP Muscle_RV12H_SP Clustal_RV12H_SP

1	0,816	0,819	0,777
2	0,902	0,922	0,898
3	0,935	0,935	0,905
4	0,938	0,923	0,905
5	0,88	0,87	0,875
6	0,923	0,941	0,896
7	0,864	0,846	0,826
8	0,945	0,917	0,915
9	0,952	0,933	0,9
10	0,863	0,907	0,888
11	0,743	0,731	0,762
12	0,59	0,61	0,586
13	0,94	0,933	0,92
14	1	1	1
15	0,934	0,931	0,913
16	0,82	0,807	0,837
17	0,915	0,883	0,905
18	0,961	0,946	0,93
19	0,872	0,883	0,87
20	0,849	0,833	0,867
21	0,75	0,935	0,871
22	0,901	0,815	0,933
23	0,883	0,866	0,838
24	0,894	0,915	0,915
25	0,777	0,765	0,71
26	0,887	0,889	0,866
27	0,946	0,931	0,918
28	0,845	0,812	0,802
29	0,965	0,951	0,955
30	0,934	0,905	0,941
31	0,807	0,805	0,765
32	0,906	0,766	0,798
33	0,798	0,776	0,752
34	0,894	0,884	0,905
35	0,931	0,928	0,852
36	0,959	0,953	0,96
37	0,847	0,81	0,776
38	0,893	0,878	0,81
39	0,919	0,916	0,879
40	0,937	0,937	0,963
41	0,689	0,634	0,69
42	0,637	0,689	0,615
43	0,948	0,913	0,859
44	0,866	0,839	0,866

Figure 6.1.2.d

	Tcoffee_RV12H_CS	Muscle_RV12H_CS	Clustal_RV12H_CS
1	0.940	0.590	0.580
2	0.400	0.790	0.740
3	0.500	0.800	0.780
4	0.560	0.650	0.570

5	0.150	0.660	0.660	
6	0.070	0.890	0.850	
7	0.440	0.690	0.680	
8	0.330	0.690	0.690	
9	0.650	0.830	0.770	
10	0.420	0.760	0.730	
11	0.100	0.400	0.480	
12	0.830	0.460	0.420	
13	0.000	0.820	0.800	
14	0.610		1.000	1.000
15	0.490	0.850	0.830	
16	0.310	0.600	0.710	
17	0.640	0.740	0.790	
18	0.250	0.910	0.880	
19	0.130	0.770	0.760	
20	0.370	0.770	0.780	
21	0.460	0.870	0.800	
22	0.490	0.670	0.880	
23	0.120	0.760	0.720	
24	0.150	0.850	0.840	
25	0.070	0.610	0.590	
26	0.010	0.650	0.590	
27	0.170	0.800	0.660	
28	0.700	0.610	0.670	
29	0.400	0.890	0.860	
30	0.230	0.790	0.880	
31	0.080	0.660	0.630	
32	0.420	0.400	0.570	
33	0.010	0.590	0.570	
34	0.180	0.790	0.830	
35	0.340	0.590	0.480	
36	0.140	0.900	0.920	
37	0.210	0.530	0.540	
38	0.500	0.610	0.480	

Figure 6.1.3.a

	T_coffee_RV30F_SP	Muscle_RV30F_SP	Clustalw_RV30F_SP
1	0.780	0.825	0.740
2	0.725	0.747	0.540

3	0.503	0.542	0.478
4	0.840	0.864	0.833
5	0.756	0.829	0.808
6	0.482	0.509	0.514
7	0.646	0.714	0.663
8	0.644	0.644	0.619
9	0.478	0.502	0.443
10	0.900	0.911	0.892
11	0.922	0.918	0.931
12	0.754	0.719	0.716
13	0.625	0.632	0.633
14	0.817	0.900	0.835
15	0.596	0.645	0.573
16	0.412	0.381	0.397
17	0.665	0.634	0.509
18	0.887	0.814	0.810
19	0.642	0.682	0.650
20	0.562	0.486	0.410
21	0.477	0.580	0.578
22	0.807	0.801	0.746
23	0.756	0.733	0.641
24	0.703	0.719	0.579
25	0.643	0.672	0.650
26	0.763	0.734	0.664
27	0.562	0.653	0.569
28	0.841	0.856	0.845
29	0.843	0.814	0.813
30	0.873	0.846	0.817

Figure 6.1.3.b:

	T_coffee_RV30F_CS	Muscle_RV30F_CS	Clustalw_RV30F_CS
1	0.160	0.310	0.340
2	0.200	0.290	0.000
3	0.030	0.020	0.030
4	0.510	0.590	0.510
5	0.270	0.340	0.380
6	0.000	0.000	0.000
7	0.000	0.410	0.000
8	0.200	0.240	0.220
9	0.000	0.000	0.000
10	0.390	0.480	0.290
11	0.520	0.670	0.670
12	0.400	0.400	0.320
13	0.210	0.200	0.120
14	0.160	0.580	0.390
15	0.350	0.350	0.380
16	0.000	0.000	0.000

17	0.310	0.340	0.240
18	0.160	0.010	0.010
19	0.210	0.230	0.260
20	0.000	0.000	0.000
21	0.110	0.040	0.090
22	0.290	0.130	0.150
23	0.270	0.000	0.070
24	0.220	0.250	0.150
25	0.000	0.000	0.000
26	0.240	0.310	0.160
27	0.150	0.200	0.270
28	0.140	0.170	0.120
29	0.520	0.460	0.480
30	0.540	0.000	0.000

Figure 6.1.3.c

	T_coffee_RV30H_SP	Muscle_RV30H_SP	Clustalw_RV30H_SP
1	0,81	0,858	0,82
2	0,764	0,778	0,775
3	0,554	0,559	0,538
4	0,868	0,911	0,851
5	0,952	0,921	0,932
6	0,648	0,74	0,827
7	0,846	0,865	0,822
8	0,651	0,64	0,625
9	0,591	0,792	0,758
10	0,915	0,935	0,908
11	0,951	0,918	0,933
12	0,758	0,715	0,716
13	0,693	0,672	0,689
14	0,835	0,901	0,842
15	0,682	0,742	0,649
16	0,58	0,786	0,75
17	0,683	0,643	0,479
18	0,919	0,928	0,901
19	0,815	0,719	0,693
20	0,679	0,724	0,743
21	0,565	0,626	0,575
22	0,819	0,801	0,807
23	0,826	0,788	0,72
24	0,79	0,782	0,719
25	0,778	0,718	0,701
26	0,773	0,783	0,743
27	0,642	0,655	0,585
28	0,867	0,884	0,867
29	0,834	0,83	0,826
30	0,868	0,921	0,841

Figure 6.1.3.d

	T_coffee_RV30H_CS	Muscle_RV30H_CS	Clustalw_RV30H_CS
1	0,25	0,33	0,3
2	0,25	0,33	0,32
3	0,05	0,06	0,07
4	0,52	0,63	0,51
5	0,47	0,54	0,58
6	0,29	0,44	0,58
7	0,56	0,59	0,52
8	0,21	0,2	0,19
9	0,11	0,35	0,43
10	0,4	0,53	0,44
11	0,66	0,68	0,68
12	0,38	0,39	0,35
13	0,27	0,24	0,27
14	0,2	0,6	0,38
15	0,17	0,57	0,39
16	0,01	0,23	0,58
17	0,31	0,38	0,11
18	0,57	0,53	0,52
19	0,39	0,34	0,31
20	0,18	0,46	0,28
21	0,11	0,05	0,11
22	0,36	0,21	0,2
23	0,36	0,35	0,2
24	0,3	0,23	0,24
25	0,06	0,16	0,18
26	0,38	0,33	0,28
27	0,36	0,37	0,31
28	0,15	0,22	0,18
29	0,51	0,52	0,55
30	0,45	0,68	0,52

Figure 6.1.4.a

	T_coffee_RV40F_CS	Muscle_RV40F_CS	Clustalw_RV40F_CS
1	0,91	0,85	0,04
2	0	0	0

3	0,67	0,76	0,71
4	0,51	0,48	0,47
5	0,68	0,7	0,64
6	0,24	0,27	0,32
7	0,32	0,34	0,23
8	0,53	0,53	0,64
9	0,43	0,4	0,47
10	0,83	0,53	0,53
11	0	0	0
12	0,54	0,55	0,67
13	0	0	0,31
14	0,39	0,41	0,37
15	0	0	0
16	0	0,01	0
17	0	0,01	0
18	0	0,51	0,62
19	0,37	0,72	0,34
20	0,66	0,69	0,69
21	0,64	0,58	0,49
22	0,52	0,46	0,08
23	0	0	0
24	0,01	0	0
25	0,69	0,68	0,72
26	0,2	0,21	0
27	0	0	0
28	0,61	0,62	0,65
29	0,35	0,22	0,35
30	0,54	0,59	0,57
31	0,98	0,94	0,98
32	0,8	0,77	0,83
33	0,46	0,56	0
34	0,19	0,21	0,24
35	0,41	0,32	0,35
36	0,52	0,64	0,63
37	0	0	0
38	0	0	0
39	0,47	0,29	0,43
40	0,51	0,54	0,51
41	0	0	0
42	0	0	0
43	0,33	0,3	0,35
44	0,35	0,33	0,44
45	0,35	0	0
46	0	0	0
47	0,51	0,29	0,4
48	0,69	0	0,72
49	0,36	0	0

Figure 6.1.4.b

	T_coffee_RV40F_CS	Muscle_RV40F_CS	Clustalw_RV40F_CS
1	0,91	0,85	0,04
2	0	0	0
3	0,67	0,76	0,71
4	0,51	0,48	0,47
5	0,68	0,7	0,64
6	0,24	0,27	0,32
7	0,32	0,34	0,23
8	0,53	0,53	0,64
9	0,43	0,4	0,47
10	0,83	0,53	0,53
11	0	0	0
12	0,54	0,55	0,67
13	0	0	0,31
14	0,39	0,41	0,37
15	0	0	0
16	0	0,01	0
17	0	0,01	0
18	0	0,51	0,62
19	0,37	0,72	0,34
20	0,66	0,69	0,69
21	0,64	0,58	0,49
22	0,52	0,46	0,08
23	0	0	0
24	0,01	0	0
25	0,69	0,68	0,72
26	0,2	0,21	0
27	0	0	0
28	0,61	0,62	0,65
29	0,35	0,22	0,35
30	0,54	0,59	0,57
31	0,98	0,94	0,98
32	0,8	0,77	0,83
33	0,46	0,56	0
34	0,19	0,21	0,24
35	0,41	0,32	0,35
36	0,52	0,64	0,63
37	0	0	0
38	0	0	0
39	0,47	0,29	0,43
40	0,51	0,54	0,51
41	0	0	0
42	0	0	0
43	0,33	0,3	0,35
44	0,35	0,33	0,44
45	0,35		0
46	0		0

47	0,51	0,4
48	0,69	0,72
49	0,36	0

Figure 6.1.5.a

	T_coffee_RV50F_SP	Muscle_RV50F_SP	Clustalw_RV50F_SP
1	0.736	0.757	0.809
2	0.409	0.341	0.286
3	0.604	0.612	0.868
4	0.956	0.949	0.894
5	0.928	0.928	0.872
6	0.658	0.627	0.741
7	0.578	0.617	0.695
8	0.821	0.845	0.863
9	0.747	0.697	0.748
10	0.698	0.742	0.817
11	0.635	0.677	0.597
12	0.668	0.724	0.881
13	0.908	0.887	0.760
14	0.857	0.817	0.716
15	0.630	0.625	0.427
16	0.814	0.717	0.814

Figure 6.1.5.b

	T_coffee_RV50F_CS	Muscle_RV50F_CS	Clustalw_RV50F_CS
1	0.240	0.400	0.040
2	0.160	0.000	0.000
3	0.240	0.320	0.710
4	0.870	0.840	0.470
5	0.700	0.730	0.640
6	0.140	0.130	0.320
7	0.040	0.080	0.230
8	0.550	0.600	0.640
9	0.130	0.000	0.470
10	0.130	0.320	0.530
11	0.160	0.190	0.000
12	0.230	0.150	0.670
13	0.660	0.590	0.310
14	0.540	0.500	0.370
15	0.140	0.040	0.000
16	0.450	0.270	0.000

Figure 6.1.5.c

	T_coffee_RV50H_SP	Muscle_RV50H_SP	Clustalw_RV50H_SP
1	0.758	0.770	0.666
2	0.505	0.455	0.355
3	0.929	0.895	0.895
4	0.929	0.895	0.895
5	0.928	0.928	0.939
6	0.720	0.678	0.668
7	0.653	0.633	0.624
8	0.776	0.804	0.857
9	0.850	0.807	0.764
10	0.734	0.777	0.739
11	0.711	0.684	0.678
12	0.782	0.754	0.719
13	0.915	0.906	0.919
14	0.873	0.839	0.824
15	0.664	0.648	0.659
16	0.797	0.809	0.660

Figure 6.1.5.d

	T_coffee_RV50H_CS	Muscle_RV50H_CS	Clustalw_RV50H_CS
1	0.340	0.380	0.260
2	0.090	0.000	0.000
3	0.720	0.690	0.530
4	0.720	0.690	0.530
5	0.710	0.720	0.760
6	0.220	0.150	0.150
7	0.010	0.120	0.030
8	0.520	0.590	0.690
9	0.280	0.100	0.000
10	0.050	0.350	0.200
11	0.270	0.290	0.260
12	0.270	0.200	0.260
13	0.700	0.660	0.700
14	0.560	0.550	0.460
15	0.140	0.150	0.170
16	0.430	0.440	0.220