ABSTRACT

CHOI, KWANGBOM. P-Coffee: a new divide-and-conquer method for multiple sequence alignment (Under the direction of Dr. Dennis R. Bahler).

We describe a new divide-and-conquer method, P-Coffee, for alignment of multiple sequences. P-Coffee first identifies candidate alignment columns using a position-specific substitution matrix (the T-Coffee extended library), tests those columns, and accepts only qualified ones. Accepted columns do not only constitute a final alignment solution, but also divide a given sequence set into partitions. The same procedure is recursively applied to each partition until all the alignment columns are collected. In P-Coffee, we minimized the source of bias by aligning all the sequences simultaneously without requiring any heuristic function to optmize, phylogenetic tree, nor gap cost scheme. In this research, we show the performance of our approach by comparing our results with that of T-Coffee using the 144 test sets provided in BAliBASE v1.0. P-Coffee outperformed T-Coffee in accuracy especially for more complicated test sets.

Keywords: multiple sequence alignment, partition wall, wall identification, wall selection

P-Coffee: a new divide-and-conquer method for multiple sequence alignment

by

Kwangbom Choi

A thesis submitted to the Graduate Faculty of

North Carolina State University

In partial fulfillment of the

Requirements for the degree of

Master of Science

In

Department of Computer Science

Raleigh, NC

January 2005

Approved by:

Dr. Jon Doyle	Dr. Subhashis Ghosal

Dr. Dennis R. Bahler
Chair of Advisory Committee

DEDICATION

This thesis is dedicated to my parents Cheeman and Byunggon, my sisters Yoonsil and Yoonjung, and my family Hyewon, Areme, and Minyoung, who have been always there for me with unconditional love.

BIOGRAPHY

Kwangbom Choi received his Bachelor's degree in Chemical Engineering from Seoul National University in Korea. He worked over five years as a chemical engineer at Hanwha Corporation. He began his involvement in Computer Science at the University of Illinois, Urbana and Champaign as a non-degree undergraduate student. He has been a Master's student at the North Carolina State University also in the Department of Computer Science, and completed a minor in Statistics.

ACKNOWLEDGMENTS

I would like to thank Dr. Dennis R. Bahler for his guidance and support during this research. I am grateful to Dr. Jon Doyle and Dr. Subhashis Ghosal for being on my advisory committee. I am also thankful to Dr. Christopher G. Healey for his help in using the systems in Knowledge Discovery Laboratory.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Cells and Proteins

Cells are the elementary units of all living organisms. Each cell has its own functions, which are mostly pursued by the biochemical activity of one or more *proteins*. Every protein is composed of a linear chain of amino acids (or *residues*) in a particular order. There are twenty different amino acids that are used to make up proteins.

When a cell needs to produce a certain protein for its proper function, it uses the genetic information encoded in the form of *DNA (deoxyribonucleic acid)* sequence within its nucleus. This DNA fragment is called a *gene*. By a specialized mechanism, a cell reads a gene and then produces (or *transcribes*) an intermediary molecular called *RNA (ribonucleic acid)*. RNA travels from the nucleus to the cell component called *cytoplasm*, where a target protein is produced (or *translated*) according to the genetic code transcribed in RNA. The

overall procedure to generate a protein is called *gene expression*. The formulated protein then moves to the suitable position in the cell, and participates in chemical reactions to accomplish the mission of the cell.

1.2 Evolution from the Standpoint of Molecular Level

The evolutionary process brings about changes in DNA sequence. One major source of such change is mutations, which are errors that randomly occur during the DNA replication. There are different kind of mutations: point mutations (alteration of one nucleotide), insertions (the gain of DNA sub-sequences), deletions (the loss of DNA sub-sequences), and other DNA mutations. Only neutral or advantageous changes, which have insignificant or favorable influence on the function of proteins, are accepted, since otherwise mutations result in death by the selection process. This implies that some sequence regions are more subject to mutation. In other words, sequence regions that play important roles in forming the structure and function of proteins (called *conserved region* or *motif*) are less likely to be mutated.

Sequences are *homologous* if they have evolved from a common ancestor. Homologous sequences are called *homologs*. Note that they usually have similar sequences but may not have similar function. Since sequence alignment shows sequence similarity but not sequence homology directly, homology can only be inferred from the degree of similarity in sequence or structure.

1.3 Phylogenetic Relationship among Proteins

As described in the previous section, proteins hold the information about the evolutionary history of an organism in their sequences. Therefore, by comparing amino acid sequences (or protein sequences) from different organisms, we can estimate the evolutionary relationships between the organisms. The evolutionary relationships among species or proteins are frequently represented using trees (called *phylogenetic trees*), in which parent nodes are ancestors of their child nodes. In more informative trees, the evolutionary relatedness among organisms is often expressed using the configuration of nodes and the length of branches.

The degree of homology also reflects the evolutionary distance between two organisms. Sequence identity (%) between two protein sequences is frequently used to estimate the degree of homology and, thus, the evolutionary distance.

$$sequence\ identity\ (\%) = \frac{the\ number\ of\ identical\ residues\ when\ aligned}{the\ number\ of\ residues\ in\ the\ shorter\ sequence} \times 100 \quad (1.1)$$

1.4 The Importance of Sequence Alignment Methods

A staggering amount of DNA sequences as well as protein sequences from various kinds of organisms became available by the success of worldwide genome projects. Since characterizing all these sequences is time-consuming and costly, knowledge-based or *ab initio* prediction techniques are commonly adopted to infer the structure and the function of proteins.

Most prediction techniques assume that protein function is determined by its three-dimensional structure, and protein structure, in turn, depends upon its sequence. Therefore, sequence-level analysis, mostly pursued by sequence alignment, has been one of the major research areas in Bioinformatics.

Homology can also help to predict protein function. In this approach, a new protein sequence is compared with other protein families (or homologs) whose structures or functions are already known. If we can identify homology between a new sequence and a protein family with known function, we can infer that the function of a new sequence would be similar to that of the homologs.

1.5 Multiple Sequence Alignment

Under the "similar sequence – similar structure – similar function paradigm", the ultimate goal shared by various sequence alignment methods is to help predicting proteins' structure or functions by identifying evolutionarily related amino acids (*residue similarity or equivalency*) among sequences. Depending on how similarities are identified, alignment methods can be classified into three categories: sequence-sequence alignment (or sequence comparison), structure-structure alignment (or structure comparison) [1], and sequence-structure alignment (or threading) [2, 3].

Since protein structure remains relatively unchanged as opposed to protein sequence during the course of evolution [4], structure-based alignment methods produce more reliable results for the detection of distant homologs even below the 20~30% sequence identity region (so-called "twilight zone") [5]. But they generally require considerable computational

resources. On the other hand, sequence comparison is relatively much faster, but does not generally identify remote evolutionary relationships below the twilight zone, although the amount of sequence data is significantly larger than that of structure. To overcome this drawback of pairwise sequence alignment, multiple sequence alignment (MSA) is usually adopted for more reliable homology detection.

MSA is to arrange potentially equivalent residues of multiple sequences in a vertical column. For some sequences that do not contain a similar residue (possibly caused by insertion or deletion), we use a *gap* (denoted by '–' in this thesis) instead. One rule here is that we have to preserve the order of residues in each sequence and all the residues must be assigned somewhere in the alignment. MSA is mainly used to identify highly conserved regions in a protein family, which can be an indication of homology. MSA is also used in protein structure classification, and prediction of protein structure and/or function [6]. MSA is a computationally difficult problem. The NP-completeness of MSA is shown by Wang et al [7].

1.6 Our Approach

In this research, we propose a new MSA method, P-Coffee, motivated by the idea that if we can effectively identify alignment columns out of a given set of protein sequences, we will eventually be able to obtain the entire alignment just by assembling the identified columns. P-Coffee is a tree-based divide-and-conquer algorithm. In this method, we first extract candidate columns from a sequence set, test each candidate column, accept only qualified ones, and place them in the right position in the overall alignment. The columns

divide the original sequence set into partitions, and this makes it possible to compute the entire alignment by the divide-and-conquer algorithm. We name the algorithm "P-Coffee" (partitioning-based multiple sequence alignment method that uses consistency-based objective function for alignment evaluation) after "T-Coffee", a most frequently used MSA tool, in that our new alignment method uses position-specific substitution scores first formulated by T-Coffee.

1.7 Outline

In Chapter 2, we describe two major common components of sequence alignment methods, an objective function and optimization strategy. A frequently used benchmark resource, BAliBASE, is also described in this chapter. In Chapter 3, we explain the principles of P-Coffee in detail, and illustrate the principles with an example. We show how the values of parameters introduced in P-Coffee are determined in Chapter 4. The benchmark result is also presented in this chapter. Finally, in Chapter 5, we summarize this research, and suggest future direction of our work.

CHAPTER 2

Background Information

Sequence alignment methods are composed of two key components. One is a scoring scheme (or an objective function) that evaluates the mutational equivalency between two residues, and the other is an optimization strategy that describes how to reach to the optimum alignment under the given scoring scheme. In this chapter, we will discuss these two constituents. Additionally, we will explain a benchmark resource proposed by Thompson et al [8], which is used widely for the performance assessment of MSA tools.

2.1 Optimization Algorithms

Sequence alignment algorithms can be classified by the notion of evolutionary relatedness among sequences: Global alignment is, as implied by the name, to align similar residues with the notion that the entire sequences are related. But for remote homologs, only

some sub-sequences may be related. Based on this notion, in local alignment, residues in such potentially related sub-sequences are aligned.

MSA algorithm can also be categorized by whether the algorithm deals with all the sequences simultaneously or not. Simultaneous alignment is a notion that describes an algorithm considering all sequences simultaneously when aligning multiple sequences. By taking the whole sequence set into account concurrently, the simultaneous approach can avoid errors made by arbitrarily fixing an optimal residue alignment for a subset of sequences. It has been reported that this approach is advantageous especially when pairwise alignment becomes unreliable due to the low similarity between sequences [9].

There exist exact methods that use simultaneous approach and find the mathematically optimum alignment. Exact methods employ generalized Needleman and Wunsch algorithm [10], the multi-dimensional dynamic programming (DP). But, unfortunately, they can handle only a limited number of sequences [6]. The computational complexity of MSA problem makes it difficult to align all the sequences simultaneously without appropriate handling of sequence set. To overcome this problem, various heuristic methods have been introduced. They roughly fall into two categories: the progressive approach and the iterative approach.

2.1.1 Progressive Methods

Progressive methods are also based on DP, but DP is used pairwise; that is, in such a way that always involves two sequences (or two sequence groups in bulk). This is made possible by aligning the most closely related sequences or groups first at any steps along the pre-computed phylogenetic tree. The advantage of progressive approaches is that they are

usually simple and fast. On the other hand, a major drawback of this approach is that once two sequences or groups are aligned into one group, the positions of residues in the group are fixed, and no more adjustments for errors introduced in earlier stages are possible. PileUp [11], MultAlign [12], ClusterW [13], MULTAL [14], PIMA [15], and T-Coffee [16] belong to this category.

2.1.2 Iterative Methods

Iterative approaches begin by making one or more initial alignments of the sequences. These alignments are generated randomly in some methods. The initial alignments are then modified to generate more optimized alignments with regard to an objective function. In each iteration (or cycle), one or more existing sub-optimal alignments are revised until they converge. Various rules are used for the revision, and iterative methods are characterized by such rules. The main shortcoming of iterative methods is that they require, sometimes prohibitively, a large amount of computational time without any promise that the optimum will be found. Stochastic methods are often adopted in the iterative approach: simulated annealing in HMMT [17], genetic algorithm in SAGA [18], tabu search [19], hidden Markov model in HMMER [20], and Gibbs sampler [21]. PRRP [22] is also a stochastic iterative method.

2.2 Objective Functions

The reliability of objective functions used in sequence comparison depends upon their ability to represent numerically the biological significance of aligned residue pairs. Because of the complexity of evolution, it is impossible to make a single universally employable

objective function that can distinguish biologically meaningful alignment in any situation. The most widely used objective functions for sequence alignment are *substitution matrices*. A substitution matrix is a two-dimensional matrix whose rows and columns are labeled by residues, and which provide a way to discern whether an aligned residue pair is plausible. Substitution scores are derived from the frequencies of point mutation observed in multiple alignments of homologous protein sequences. So if a residue pair has a higher score in a substitution matrix, the mutation between the corresponding two residues is more likely to occur. Substitution matrices are usually given in the form of 20×20 matrix, covering every possible pair of existing amino acids. The most naïve substitution scoring scheme is to give score of 1 to the exact match, and 0 to an unmatched pair. This is called the *identity matrix*. Many more realistic substitution matrices have been reported, but only two matrices, PAM and BLOSUM, are most commonly employed in modern sequence alignment.

2.2.1 PAM Matrices

PAM (percent accepted mutation) [23] matrices are derived from the number of mutational events observed throughout the analyzed sequences of a specific evolutionary distance. The evolutionary distance is measured in percent divergence (or 100 – sequence identity) of sequences compared. So, 1 PAM denotes that it is obtained by comparing sequences with over 99% sequence identity. Each PAM matrix can only be used for sequences in a specific evolutionary distance, but the distance can be computationally extended by multiplying a matrix by itself, under certain independence assumptions.

2.2.2 BLOSUM Matrices

BLOSUM (blocks of substitution matrix) [24] matrices are the most frequently adopted objective function in sequence comparison today. Similarly to PAM matrices, they are empirically derived from the frequency observation of residue substitutions, but the difference from PAM matrices is that they count only point mutations that occur in highly conserved regions in many protein homologs. Since these regions are more likely to indicate biologically meaningful relationships among residues, the residues are more likely to be aligned correctly, and the substitution scores are relatively more reliable[25]. As in PAM matrices, each BLOSUM matrix is derived from a different evolutionary distance. In BLOSUM, the evolutionary distance is measured in sequence identity (%) of the sequences compared. For instance, BLOSUM62 matrix, commonly used in BLAST, is derived from many protein families containing sequences of 62% identity on average.

2.2.3 Consistency-based Objective Function

Instead of depending solely on a single substitution matrix as explained above, position-specific residue pair scores are derived from the collection of the aligned pairs obtained by other pairwise alignment methods. Consistency-based objective functions evaluate the *consistency* among various pairwise alignment results [6, 16, 26]. T-Coffee is a progressive method that uses the position-specific scoring scheme.

By default, T-Coffee first carries out two independent pairwise sequence alignments for every possible pair of the sequences at hand: one by global alignment with the Needleman and Wunsch algorithm (NW), and the other by local alignment with the SIM algorithm [27]. All and only aligned residue pairs in each pairwise alignment are registered

in separate base list (called the primary library in T-Coffee), with the sequence identity (%) as their preliminary weights (or scores). For the local alignment library, the ten highest scoring local alignments are accepted. Next, the union is taken over all the residue pairs listed in the NW and SIM primary libraries to construct a single primary library. For residue pairs that exist in both libraries, the preliminary scores are simply added. Then, T-Coffee gives additional weight to the residue pairs that can be linked by another residue contained in the remaining sequences. For each residue pair in the primary library, all the remaining sequences are examined in search of such linkage. Whenever a link residue is found, the smaller weight of either linkage is added to the current weight. This process is called *library extension*. The final residue score can be expressed as the following equation.

$$\begin{aligned} &\textit{the final residue pair score between } Seq_1\text{-}R_1 \textit{ and } Seq_2\text{-}R_2 \\ &= \sum_{b=1}^{m} \textit{preliminary weight}(Seq_1\text{-}R_1,\ Seq_2\text{-}R_2) \\ &+ \sum_{\substack{i=1 \\ i \neq Seq1 \\ i \neq Seq2}}^{n} \sum_{j} \min\{\textit{weight}(Seq_1\text{-}R_1,\ Seq_i\text{-}R_j),\ \textit{weight}(Seq_i\text{-}R_j,\ Seq_2\text{-}R_2)\} \end{aligned} \qquad (2.1)$$

where m denotes the number of (base) primary libraries, n is the total number of sequences, and Seq_i-R_j means residue j in sequence i.

With these position-specific substitution scores, T-Coffee then performs a progressive alignment. A distance matrix and a neighbor-joining guide tree (a kind of phylogenetic tree) are obtained using the scores in the extended library, and, finally, the alignment solution is

computed by a round of two-dimensional dynamic programming according to the order implied by the guide tree configuration. The sum-of-pairs score is used as an objective function in dynamic programming. The whole process is illustrated in Figure 2.1.

The main advantage of this position-specific scoring scheme is that we can plug in any pairwise alignment result to form a primary library. Thanks to this feature, we can easily incorporate various information into the substitution scores, in the expectation that this will raise the reliability of the scores.

Although T-Coffee extended library holds every single signal of residue equivalency in a given sequence set, T-Coffee has left room for improvement in accuracy because of its dependence on a progressive approach while constructing an alignment solution. This was a reasonable decision, since iterative approaches may incur prohibitive computational cost without promise of a better solution. At the same time, however, it is worth looking for a method that fully leverages the power of the T-Coffee extended library. In this work, we suggest an improved way of using the T-Coffee extended library to align multiple protein sequences.

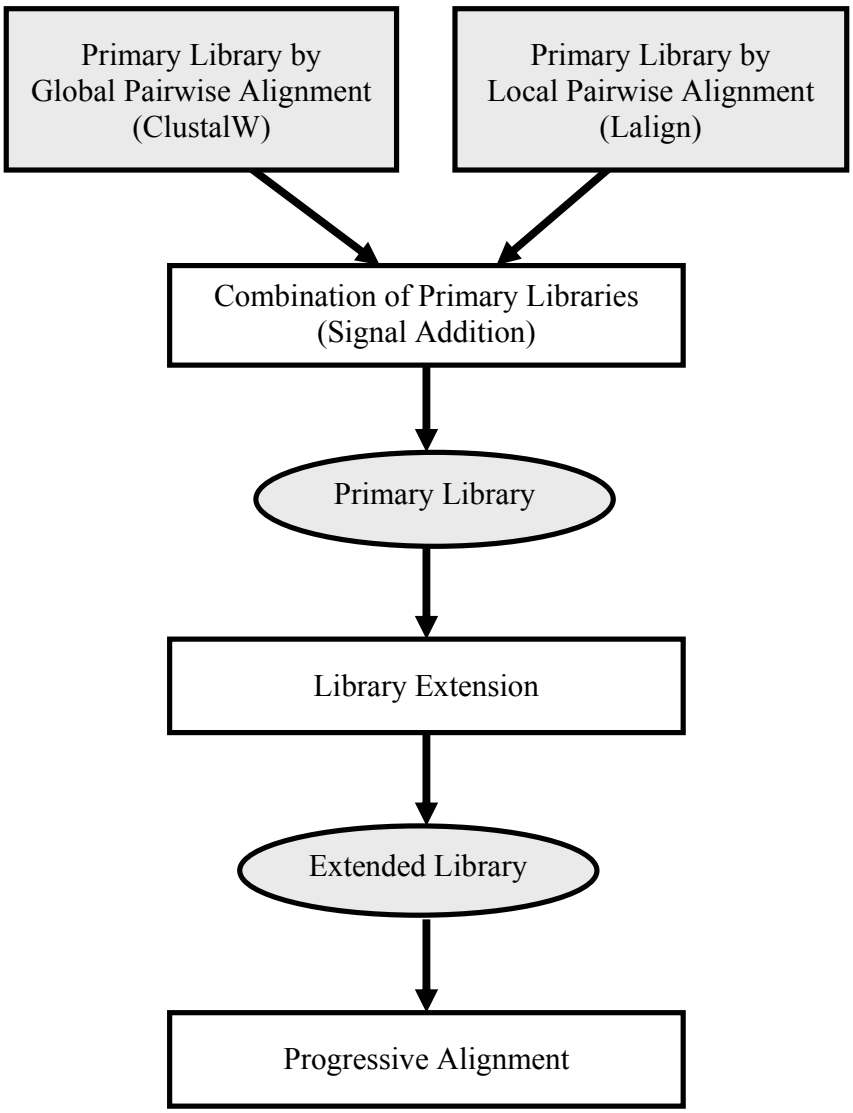


Figure 2.1: The flow of T-Coffee. This figure is redrawn from the author's version. [16]

2.3 BAliBASE

BAliBASE (benchmark alignment database) v1.0 provides 144 accurate reference alignments so that we can easily benchmark the performance of MSA methods. The sequence sets are categorized by length and sequence identity (%), compositional divergence, and the presence of orphan sequences (a sequence that has lower than 25% sequence identity with other members in a set), internal insertions, and N/C-terminal extensions (insertion of sub-sequence at the beginning or at the end of original sequences). The core blocks (or conserved regions) in a sequence set are capitalized and marked with underlines so that we can see whether a test MSA method can catch biologically significant signals of residue equivalencies.

Table 2.1: The description of BAliBASE categories

		Sequence identity	Number of sequences range (average)	Number of sequence sets	Description
Reference 1	V1	< 25%	3~5 (4.3)	27	sequences with similar identities and lengths
	V2	20~40%	4~6 (4.4)	27	
	V3	> 35%	4~6 (4.7)	28	
Reference 2		12~48%	15~24 (19.6)	23	up to 3 orphans
Reference 3		19~41%	19~28 (22.4)	12	divergent families (up to 4 sub-groups)
Reference 4		13~43%	4~17 (8.4)	15	N/C-terminal extensions (up to 400 residues)
Reference 5		19~38%	5~19 (9.3)	12	internal insertions (up to 100 residues)

BAliBASE also provides a standard program, BaliScore, that evaluates accuracies of the MSA results. The program computes two different scores for each alignment: one is sum-of-pairs score (or SP score) that estimates the ability to identify correct residue pairs, and the other is column score (or TC score) that assess the competence in correctly aligning the entire column. The two scores are calculated by the following equations.

$$SP\ Score = \frac{\sum_{i=1}^{Mt} t_i}{\sum_{j=1}^{Mr} r_j} \tag{2.2}$$

where $t_i = \sum_{\substack{u=1 \\ u \neq v}}^{N} \sum_{v=1}^{N} p_{iuv}$, $r_j = \sum_{\substack{u=1 \\ u \neq v}}^{N} \sum_{v=1}^{N} p_{juv}$ each for test and reference alignment,

$p_{iuv} = \begin{cases} 1 \text{ if residues } R_u \text{ and } R_v \text{ in the i-th column are aligned} \\ 0 \text{ otherwise,} \end{cases}$

N is the number of sequences contained in the problem set

M_t is the number of columns in test alignment, and

M_r is the number of columns in reference alignment.

$$TC\ Score = \frac{\sum_{i=1}^{Mt} c_i}{M_t} \tag{2.3}$$

where $c_i = \begin{cases} 1 \text{ if all residues in the i-th column are aligned in reference} \\ 0 \text{ otherwise,} \end{cases}$

The performance (SP scores only) of other well-known MSA methods are available in the website http://www-igbmc.u-strasbg.fr/BioInfo/BAliBASE/prog_scores.html [28]. But T-Coffee outperformed all the other methods on average, so, in this research, we will extensively examine the performance of our new alignment method by comparing both the SP score and TC score with those of T-Coffee, using the test cases provided in BAliBASE v1.0.

CHAPTER 3

The Principles of P-Coffee

In this chapter we explain the operation of our approach, P-Coffee. Specifically, we show in detail how partition walls are identified from a given set of sequences, and how P-Coffee aligns sequences with identified walls. An illustrative example of identifying a partition wall is also given.

3.1 Identification of Partition Walls

Identifying partition walls from an unaligned mess of residues is the most important component of the P-Coffee algorithm. In this section, we will make clear the definition of partitions and partition walls, and will introduce the concept of complete and virtual walls. We will also show the entire procedure of identifying partition walls, and the issues related to the optimization of the procedure.

3.1.1 The Definition of Partitions and Partition Walls

A *partition wall* is defined as a set of residues, each of which is chosen from the given sequences. At most one residue can be chosen from each sequence. If no residue is chosen from a sequence, a gap is used in its place. A partition wall exactly corresponds to a column in a candidate alignment solution for a given sequence set. A *complete* wall refers to one that does not contain any gaps. Two walls are *compatible* if they are completely separated and do not cross over each other. Otherwise, two walls are said to be *conflicting*. The residues in a partition wall are sorted in the order of sequence id (usually a unique integer assigned to each sequence) so that we can easily check the compatibility of two walls.

A *partition* is a set of sub-sequences between two complete walls. A sub-sequence can be a null string if no residues of a certain sequence are involved in the partition. A partition cannot include any walls. The *partitioning* refers to dividing a given sequence set into multiple disjoint partitions sectioned by single or multiple partition walls.

Lots of complexity is caused by allowing gaps in a wall. A partition cannot be obtained with incomplete walls because there is no way of determining which position a gap belongs to. We have overcome this difficulty by introducing the concept of *virtual* wall. To obtain a partition between incomplete walls, we build virtual walls by borrowing a residue from the appropriate neighboring walls that contain a residue for the corresponding sequence. For a gap in the left wall of a partition, we borrow a residue from the nearest wall to the left with no gap for the same sequence. Correspondingly, for a gap in the right wall, a residue is borrowed from the nearest wall to the right. Residue donors do not have to be complete, but have to have a residue for a required sequence. Virtual walls are always complete. But if we use virtual walls for partitioning, partitions are no longer disjoint. This means that they can

share some sub-sequences, which will be divided when more walls become available in later steps. Note that virtual walls cannot be used to construct the entire alignment, because they do not exist in reality.

In the case illustrated in Figure 3.1, there are two complete walls {S, D, R, Q} and {K, K, S, A}, which are placed in solid boxes. There are four incomplete walls, which contain underlined residues: {V, −, T, A}, {S, −, A, G}, {G, S, −, S}, and {R, −, K, K}. Gaps are denoted by '−'. Walls are aligned because they correspond to the columns in some alignment, as mentioned earlier.

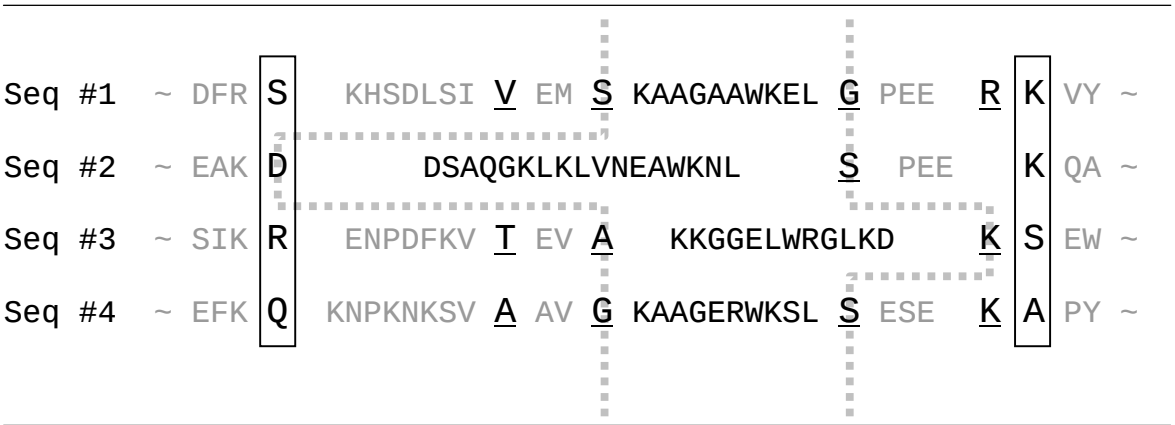


Figure 3.1: An example of virtual walls (sequences from 1aab_ref1 in BAliBASE v1.0)

Suppose we want the partition between the walls {S, −, A, G} and {G, S, −, S}. Because both of the walls are incomplete, we have to construct virtual walls to obtain the partition. In the left wall {S, −, A, G}, the gap for sequence #2 should be replaced with the residue in the nearest left wall with no gap for sequence #2. In this case, because {V, −, T, A} had again a

gap for sequence #2, we had to borrow a residue from {S, D, R, Q}, one of the complete walls. Thus the left virtual wall is {S, D, A, G}. Similarly, for the gap in the right wall {G, S, −, S}, we fill the gap for sequence #3 with the residue in the nearest right wall {R, −, K, K}. Note that {R, −, K, K} is not complete, but holds a residue for sequence #3. Thus the right virtual wall is {G, S, K, S}. The resulting partition is, therefore, {KAAGAAWKEL, DSAQGKLKLVNEAWKNL, KKGGELWRGLKD, KAAGERWKSL}.

3.1.2 The Advantages of Partitioning

Partitioning a set of sequences at hand has two major advantages.

First, it reduces the individual problem size by dividing the original sequence set into multiple independent sets of sub-sequences. Once a partition is fixed, we do not need to consider alignment between a residue inside the partition and others outside the partition. This significantly reduces the complexity in aligning the sub-sequences in a certain partition, when compared with aligning without partitioning. The most important point here is that the problem complexity reduces to O(L), linear time in the (average) length of the sequences, assuming the number of sequences is fixed. Some partitions will contain null strings as partitioning advances. For these partitions, we are only concerned with the remaining sequences. Because the running time of the depth-first tree search algorithm is $O(B^d)$, where B is the branching factor and d is depth, the reduced depth of the problem will also result in considerable speedup. We can obtain the overall alignment solution by simply "concatenating" the alignment of each partition and partition walls in the original order provided in the sequence set.

Second, we can fix a part of the solution by identifying, evaluating, and accepting

qualified partition walls during the running of the algorithm. As mentioned in the previous section, a partition wall corresponds to a column in the alignment solution. Thus, to reach an alignment solution, we can pay our attention only to unaligned portions, which are partitions.

3.1.3 Steps to Identify a Partition Wall

Step 1. *Generate a random hierarchy*

As mentioned in Chapter 2, T-Coffee aligns sequences using a progressive approach that relies on multiple running of dynamic programming in the order implied by a precomputed phylogenetic tree. Instead of depending upon such a "hypothetical" tree, P-Coffee generates a random hierarchy, which defines the contingent rank of each sequence. This hierarchy is used when P-Coffee constructs a tree, and coordinates the parent-child relationship among residues. Some of possible random hierarchies are shown in Figure 3.2. In this four-sequence case, there exist 4! = 24 different hierarchies.

Depth 0	Sequence #3	Sequence #1	Sequence #2	Sequence #4
	|	|	|	|
Depth 1	Sequence #4	Sequence #4	Sequence #1	Sequence #3
	|	|	|	|
Depth 2	Sequence #1	Sequence #3	Sequence #4	Sequence #2
	|	|	|	|
Depth 3	Sequence #2	Sequence #2	Sequence #3	Sequence #1

Figure 3.2: Examples of random hierarchy for 4-sequence case

Step 2. *Build a tree beginning with a residue randomly picked from the depth-0 sequence*

Given a random hierarchy, P-Coffee starts building a tree by choosing one residue from the depth-0 sequence. Beginning with the chosen residue as the root of the tree being constructed, P-Coffee recursively attaches the residues that are contained in the T-Coffee library when paired with the current parent residue. The simple rule here is that residues in depth-(i) are rendered to be parent nodes of ones in depth-(i+1). In other words, edges are directed in increasing order of depth, from lesser to greater. A tree is built in depth-first fashion, and, when there are no residues that can be added to the current node, the tree extension can stop even before it reaches the maximum depth.

Step 3. *Do the depth-first search for the highest-scored path in the fully-grown tree*

After a tree has been fully grown, P-Coffee searches for the highest-scored path in the tree. When a path does not reach to the maximum depth, we can just assign a gap for each remaining sequence with depth greater than that of the leaf. Then, if we sort the nodes in the path in the order of sequence id, we can obtain a partition wall. Thus, by searching for the highest-scored path in the tree, we can obtain the highest-scored partition wall. Here we evaluate each path by sum-of-pairs score instead of path weight. Under the sum-of-pairs scoring scheme, the score of an identified partition wall is calculated by summing up the substitution scores of all the possible residue pairs. Therefore, if residues contained in a wall are more interrelated, a higher score will result.

3.2 Techniques to Speedup the Wall Identification Process

In order to speedup the identification process of partition walls, we use several tricks during the tree construction procedure. The tricks used here are not new, but helpful in making this recursive procedure affordable.

3.2.1 Combination of Tree Building and the Highest Scorer Searching

Because trees are needed only to obtain the highest-scored path, we can speed up the identification process by combining the tree construction and the highest-scored path search. Specifically, we update the position of current highest-scorer as well as its score whenever a node is added to the tree. In this way, we get the highest-scored path as soon as we finish constructing a tree. Trees are discarded without being stored in order to save memory.

3.2.2 Thresholding

Even if a residue found as a child lies within a partition range, we may not want to add it because we regard the entailed pair score to be unreliable and presumably safely neglected. The objective of thresholding is “not” to add a node that is expected to be unreliable as a member of a path being extended. But of course, if the threshold pair score is set too high, the algorithm will branch away from a node that is required to successfully construct a wall with maximum available reliability.

As explained in Chapter 2, T-Coffee uses sequence identity (%) of two sequences as the base substitution score for aligned residue pairs in the sequences. During the library extension process of the T-Coffee, such base scores are only augmented according to the

consistency of each residue pair. Therefore, sequence identity is the minimum score that is guaranteed to any aligned residue pairs in T-Coffee extended library. In addition, it is reported that sequence comparison detects relationships between protein residues reliably down to about 30% sequence identity [5]. For all these reasons, it is expected that the appropriate threshold score can be determined near the range of "twilight zone" identities.

Under the sum-of-pairs scoring scheme, one additional child node brings in multiple pair scores. As shown in Figure 3.3 by the bold bi-directional arrows, we consider all possible pairs between the child node at hand and the predecessor nodes in the path being constructed.

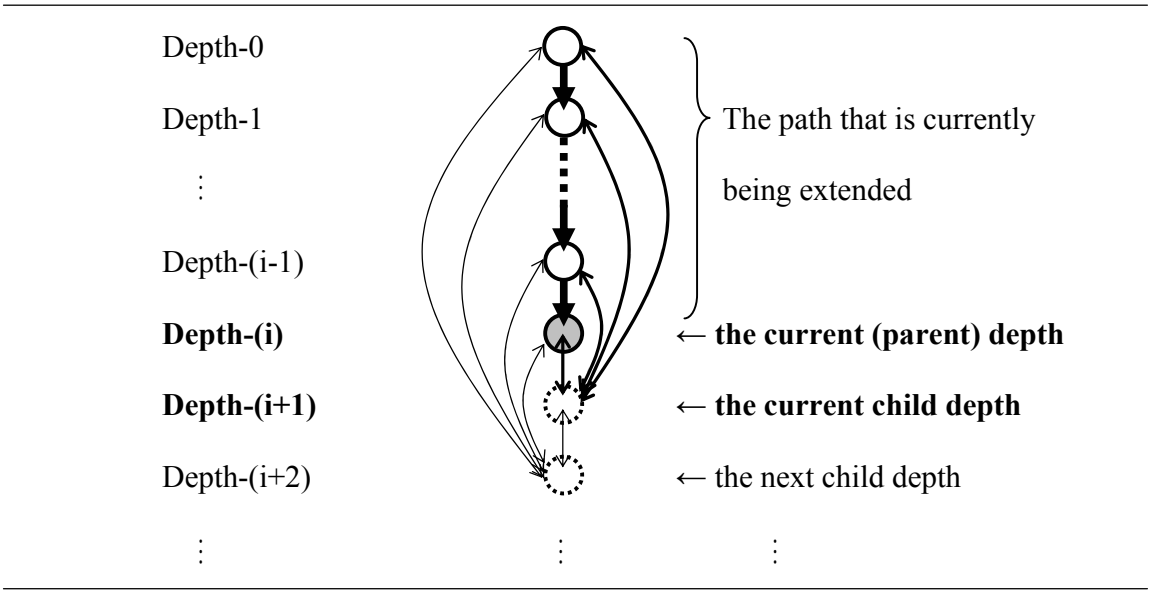


Figure 3.3: The number of new residue pairs generated by adding a child

We apply a threshold pair score to the average of all the pair scores generated by adding a child node. In this way, a child node that has fewer but stronger connections with its predecessors can also be accepted.

3.2.3 Pruning

We can further save resources (computation time and memory) by "not" adding a child if no relevant future path can possibly outscore the current highest. The maximum substitution score can be trivially found in the T-Coffee extended library. And, we also know the remaining depths to go and, therefore, the maximum number of residue pairs that will be brought in under the sum-of-pairs scoring scheme. Therefore, by giving each residue pair the maximum substitution score in the T-Coffee library, we can easily compute the upper bound on the score that can be achieved by the path that is currently being extended. By comparing this estimated upper bound with the current highest that is previously found, we can decide whether to add a child node or not.

3.2.4 Sorting of Candidate Child Nodes

To maximize the effectiveness of pruning, it is better to come up with as high as possible a score in earlier stage of the tree construction. To promote this situation, we designed our method to consider pairs in decreasing order of score when examining candidate child nodes for attachment. With this local greedy approach, the residue that entails the highest substitution score available will be added first at each step of depth-first tree construction, making higher scoring path more probable in earlier stage of tree construction.

3.2.5 Jumping

Jumping describes finding related residues from lower depths than the current child depth. Since with jumping more candidate residues are within the scope of a parent node, this modification considerably increases the average branching factor. Thus, more candidate paths (partition walls) are examined while searching for the highest-scored path. There is a trade-off between the thoroughness of searching and the resulting computational cost.

Although a more branched tree would result, jumping does not mean that we can necessarily identify a more reliable path from the tree, because of a side effect incurred by jumping: jumping introduces a gap for the skipped sequence. For example, the one step jumping from node (a) to node (c) will result in the path {…, a, −, c, …} as shown in Figure 3.4.

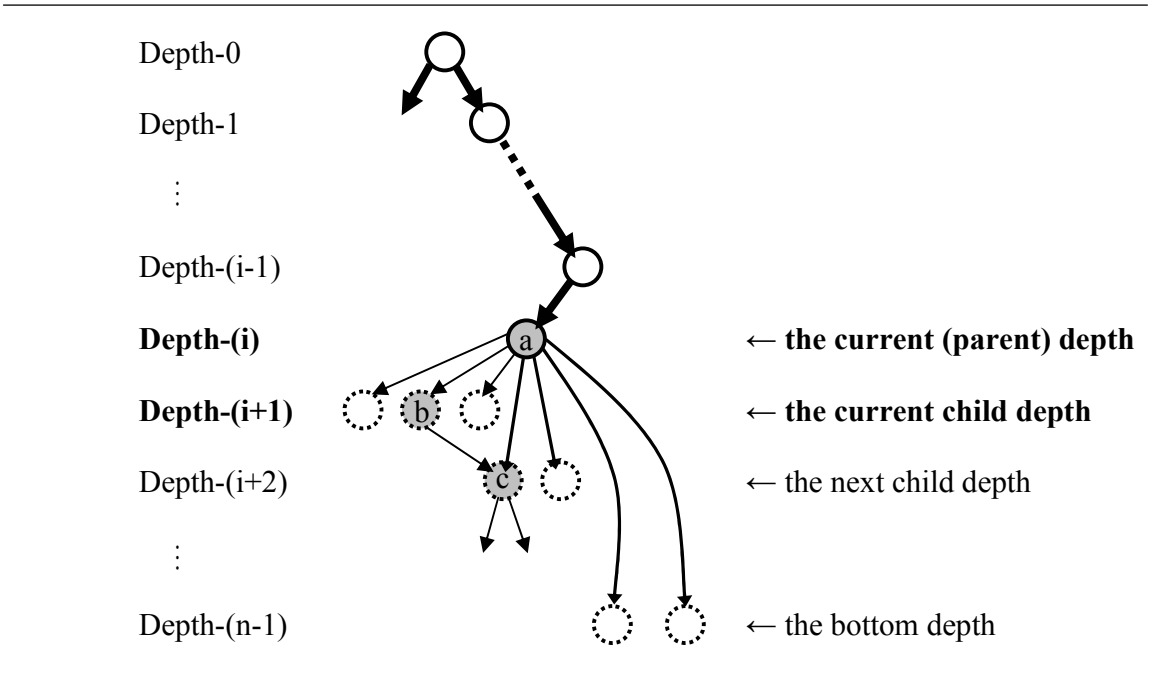


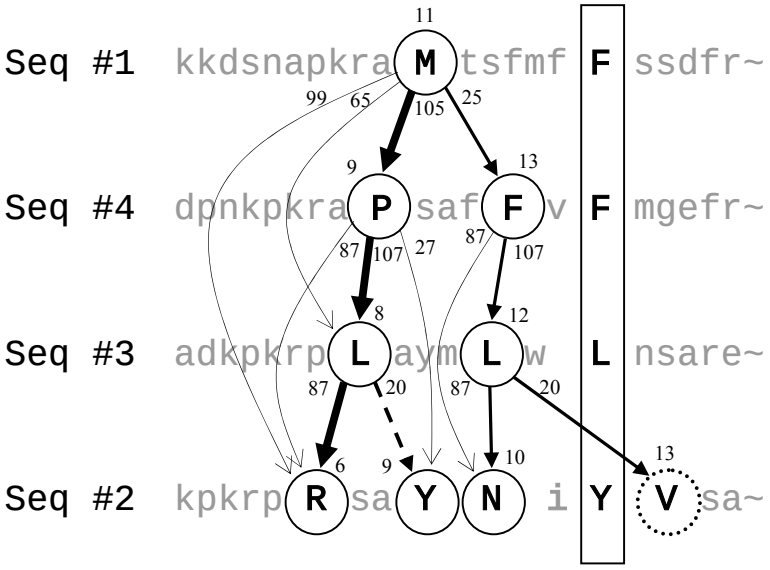
Figure 3.4: Jumping

Dotted circles denote candidate nodes that may be added to the current (parent) node, which is currently the node (a). Since a gap significantly reduces the sum-of-pairs score of a wall by negating maximum (n – 1) substitution scores, where n denotes the total number of sequences, a path with jumping history is less likely to make the highest score in the tree. This can be easily noticed by comparing the score of the path {…, a, b, c, …} with that of the path {…, a, −, c, …}. The latter will lose every pair score that involves the node (b), under the sum-of-pairs scoring scheme.

Therefore, if the partition at hand contains many complete or almost complete walls, jumping will not bring any improvement in accuracy. On the other hand, jumping will expedite the tree building process when it is used for a partition that contains walls with many gaps, because we do not have to wait until all non-gaps are placed in the top depths consecutively by a random hierarchy. To reflect this idea, we designed P-Coffee to jump only in case we cannot find any candidate child from the current child depth. As you may see in Section 3.4, this *lazy* jumping is allowed after lots of complete walls are extracted out.

3.3 Example of Identifying a Partition Wall

Figure 3.5 shows the 1aab (reference 1) case given in BAliBASE v1.0. Suppose a random hierarchy #1→ #4→ #3→#2 is generated, and 1-11 (residue #11 of sequence #1) is randomly picked as the root of the tree that is going to be constructed.



The sum-of-pairs score of each path

{M, P, L, R}: 105 + 65 + 99 + 107 + 87 + 87 = 550
{M, P, L, Y}: 105 + 65 + 0 + 107 + 27 + 20 = 324
{M, F, L, N}: 25 + 0 + 0 + 107 + 87 + 87 = 306

An example score calculation of a jumping path

{M, P, −, Y}: 105 + 0 + 0 + 0 + 27 + 0 = 132

Related residue pair scores

# 1 2		
11	6	**99 (*)**
# 1 3		
11	8	**65**
# 1 4		
11	9	**105**
11	13	**25**
# 2 3		
6	8	**87**
9	8	**20**
10	12	**87**
13	12	**20**
# 2 4		
6	9	**87**
9	9	**27**
10	13	**87**
# 3 4		
8	9	**107**
12	13	**107**

Figure 3.5: An example of tree construction. The path {F, F, L, Y} in a solid block denotes a previously identified complete wall, by which a partition is formed on its left side. Jumping is not allowed. The terms in the sum-of-pairs scores are given in the order of (1,4), (1,3), (1,2), (4,3), (4,2), and (3,2), where (i, j) denotes the pair of sequence id's. The table of substitution scores is given as is in T-Coffee extended library.

The relevant residue number is put on the top of each node. And, the substitution scores are placed at the bottom of each node, near the starting point of each directed edge. The relevant substitution scores are excerpted from the T-Coffee extended library that is computed beforehand for this sequence set. The lines with # sign specify the sequence numbers. Other lines indicate residue pair by their position indices in each sequence, and its score (bold face). For example, the line marked (*) means that the substitution score between 1-11 and 2-6 is 99.

A tree is built starting from the randomly picked root 1-11. So the current ongoing path is {1-11}. By the random hierarchy, sequence #4 is at the current child depth. So we refer to the category [# 1 4] in the T-Coffee library, and find out that 1-11 is related to 4-9 and 4-13, whose scores are 105 and 25 respectively. Because 4-9 makes higher score with 1-11, it is first checked for attachment. Suppose we set the threshold pair score to 10. Because the average score between 4-9 and every node in the ongoing path {1-11} is 107 / 1 = 107 > 10, it clears threshold condition. Next we check the pruning condition. From the current child depth, there are two more depths to go, so maximum 2 + 3 = 5 residue pairs would be brought in until the path reaches to the bottom depth. Because the maximum pair score is found to be 107 for this set, the upper bound score the path {1-11, 4-9} can achieve is 107 + 107 × 5 = 642. This outscores the current highest, which is zero. So the candidate 4-9 also clears the pruning condition, thus it is added to the root 1-11. The current ongoing path becomes {1-11, 4-9}. The current highest is reset to node 4-9 with the score of 107.

This simple *checking and adding* process is repeated in depth-first fashion until we have no more nodes to attach. The tree construction ends up with three paths; {M, P, L, R}, {M, P, L, Y}, and {M, F, L, N}. The sum-of-pairs scores are also calculated in Figure 3.5.

According to the score, the path {M, P, L, R} is the highest-scored path. By sorting the nodes in the path in the order of sequence id, the partition wall {M, R, L, P} is obtained.

There are a few more points to note in Figure 3.5. First, 2-13 (denoted by the dotted circle) could not be added because it is outside the partition. Secondly, when 2-9 is checked for the thresholding condition before being added to 3-8 (denoted by the dashed arrow), the average pair score between 2-9 and every node in the path {1-11, 4-9, 3-8} is computed to be $(0 + 27 + 20) / 3 \approx 16 > 10$. If we set the threshold pair score to 20, then 2-9 will not be accepted as a child of 3-8. Finally, if jumping is allowed, more parent-child relationships will be added (denoted in thin directed edges) in the tree. One example score of a path with jumping history is also calculated in Figure 3.5. The jumping paths in this case do not bring any difference except the increased computational cost, since the partition contains complete paths with high scores.

3.4 Partition Wall Selection for Alignment Solution Construction

The reliability of an alignment solution depends upon that of columns used to construct the alignment. Since columns correspond to partition walls in P-Coffee, it is crucial to correctly evaluate the reliability of the walls found during the wall identification procedure. In this section, we propose an additional criterion other than the sum-of-pairs score to select reliable walls from the pool of identified walls.

3.4.1 Reliability of a Partition Wall

There are some issues about the *reliability* of partition walls. What is the reliability of a partition wall, and how can we evaluate it? Does a higher sum-of-pairs score mean a more reliable partition wall? Is the score the only criterion for reliability we can depend upon?

As for reliability, we can simply state that reliable walls should be able to catch biologically meaningful residue equivalency. In fact, discovering correct residue equivalencies from given protein sequences is the essence of all sequence alignment tools, in that structure or function of proteins are predicted based on these equivalencies. On the other hand, evaluating residue equivalency is not as straightforward as the objective itself, because of the lack of dependable arrangements to incorporate information that relates residues into sequence-level analysis. There have been efforts to exploit other source of equivalency information such as structural data [29]. But, coming up with a sound objective function to evaluate residue equivalencies is still an important issue.

The position-specific scoring scheme such as the extended library of T-Coffee is one good example of an objective function evaluating the similarity of residues. But, what makes problems complicated is that a higher sum-of-pairs score does not always result in a column

in the reference alignment. We found that walls falling in only upper 15% of score range can be used safely to construct alignment solutions.

Facing this challenge, we paid an attention to an important point: the sum-of-pairs scoring scheme reflects the *interrelatedness* among residues contained in a column (or a wall). Based on the idea, we invented a test that detects more reliable walls from the pool of identified walls.

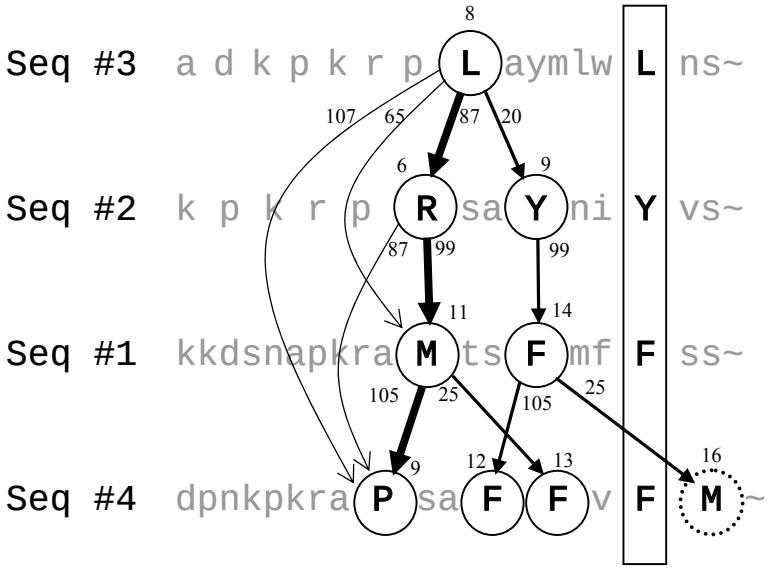
3.4.2 The True Power of Random Hierarchy

The breakthrough for the difficulty in reliability evaluation is to accept walls that appear more than some specified number of times after a sufficiently large number of iterations of wall identification. As explained in Section 3.1.3, P-Coffee first generates a random hierarchy, and arbitrarily chooses one residue from the depth-0 sequence as root to be extended. In each iteration, a different sequence hierarchy as well as a root is used, and extended fully to a different tree. During this iterative procedure, a certain wall may be identified more than once if residues contained in the wall are closely interrelated. Therefore, by accepting *reproducible* walls, we can mimic the idea of the sum-of-pairs scoring scheme. The advantage of this test is that it does not depend upon any scoring metric, so even relatively low scorers may be accepted.

As we show in Chapter 4, this approach significantly raises the accuracy of identified walls. With the combined use of the sum-of-pairs score and the reproducibility, we can also raise the accuracy of the alignment solution. Taking advantage of the statistical behavior of random hierarchy, we were able not only to make the algorithm independent from

hypothetical phylogenetic trees but also to provide a novel way to discover the interrelatedness among residues.

This can be illustrated by comparing Figure 3.5 and Figure 3.6. It can easily be noticed by directed edges that the residues in the highest-scored path {M, P, L, R} are interrelated in Figure 3.5. In Figure 3.6, we use a different hierarchy #3→ #2→ #1→#4 and a different root 3-8, and found the different highest scored path {L, R, M, P}. But as indicated in Figure 3.7, these two paths imply the same wall, {M, R, L, P}.



The sum-of-pairs score of each path

{L, R, M, P}: 87 + 65 + 107 + 99 + 87 + 105 = 550
{L, R, M, F}: 87 + 65 + 0 + 99 + 0 + 25 = 276
{L, Y, F, F}: 20 + 0 + 0 + 99 + 0 + 105 = 224

Related residue pair scores

# 1 2		
11	6	**99**
14	9	**99**
# 1 3		
11	8	**65**
67	8	**28**
# 1 4		
11	9	**105**
11	13	**25**
14	12	**105**
14	16	**25**
# 2 3		
6	8	**87**
9	8	**20**
# 2 4		
6	9	**87**
9	9	**27**
# 3 4		
8	9	**107**

Figure 3.6: Another random tree for the Section 3.3 example

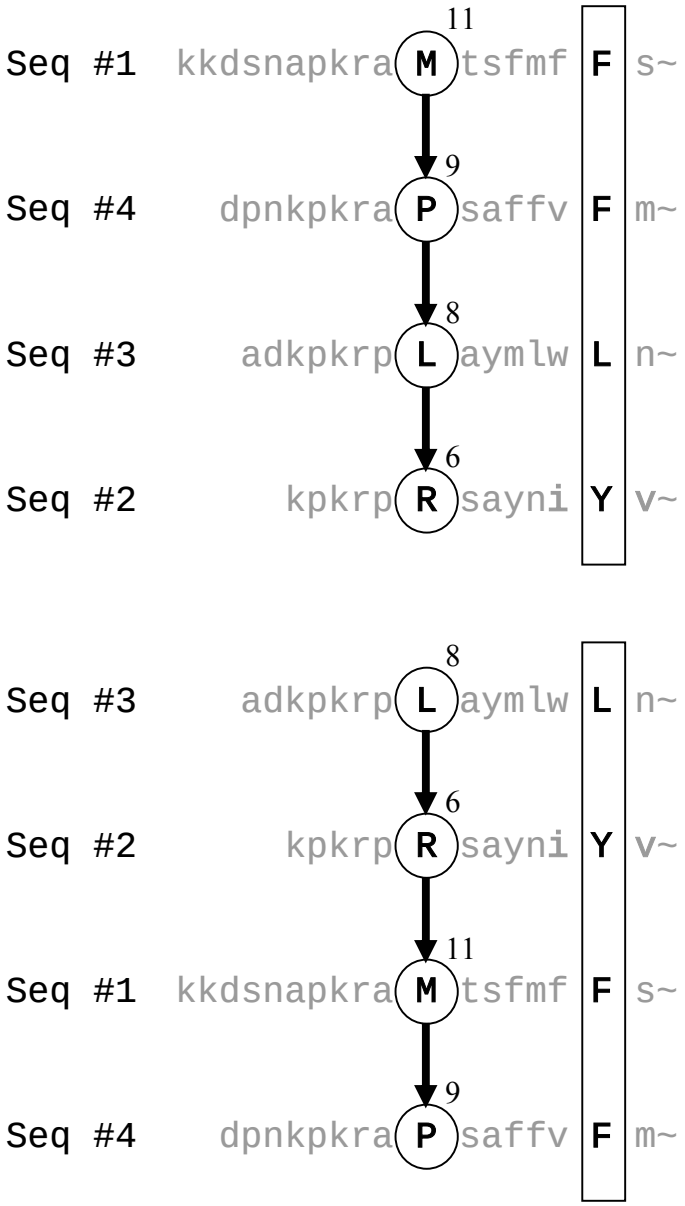


Figure 3.7: The wall identified by two different random trees

3.4.3 The Construction Procedure of Alignment Solution

The target alignment is constructed by recursively partitioning the given sequence set. There are two base cases available for this recursive procedure. If a partition is composed of at most one residue for each sequence, P-Coffee fixes the partition as one partition wall. If a partition contains only residues that belong to a certain sequence, P-Coffee separates them into distinctive columns that contain only one residue. After all partitions are fixed, P-Coffee concatenates them into one big alignment solution. To save memory, we implemented this recursive procedure so as to keep only selected walls and to handle each partition between two walls one by one. Every partition from the leftmost one to the rightmost one undergoes further partitioning during one *phase.*

There are parameters to be determined before running P-Coffee. The threshold pair score is a parameter for screening out unreliable residues in order to speed up the wall identification process. The threshold score of 10 is used throughout the phases except the last one. The *Multiplier* is a parameter that defines the number of iterations for each phase. If it is "×2", the wall identifying procedure is repeated twice the maximum sequence length number of times in the partition at hand. The *Acceptance Qualification* describes which walls to accept in each phase. If it is "double+", we only accept walls that are identified more than once. We also have to decide whether to allow jumping. Detailed description of the parameters is given in the next section.

Once a partition wall is selected, it is used not only as a building block for an entire alignment solution but also as a boundary between partitions. As explained in the previous sections, the original sequence set is divided into conceptually disjoint problem sets by partition walls. Note that we do not consider alignments of residue pairs that have a partition

wall between them. Therefore, it is important to select more reliable walls in the earlier stages, so that partitions can safely disregard outsider residues during the iterative partitioning process. Based on this idea, we designed P-Coffee to work in four phases. The values of the parameters are summarized in Table 3.1.

Table 3.1: Phases in P-Coffee. Jumping is not a parameter for the algorithm, but included in this table to show when it is activated. In phase #4, the acceptance qualification gives the priority in the order of double+'s, and then, the high scorers.

	Multiplier	Jumping	Threshold	Acceptance Qualification
PHASE #1	×1	not allowed	10	top 5% scorers
PHASE #2	×2	not allowed	10	triple+
PHASE #3	×4	lazy jumping	10	double+
PHASE #4	×2	lazy jumping	0	double+ all non-conflicting leftovers

In Phase #1, jumping is not allowed because it can be safely assumed that the sequence set should contain many complete or almost complete walls initially. Only the top 5% highest scorers can survive in this phase. Considering that one partition wall reduces the size of the corresponding partition to half on the average, we can still divide the original problem into multiple partitions of significantly reduced size even retaining only this small

portion of all identified walls. The wall identification is repeated the maximum sequence length number of times. The reproducibility criterion is not applied in this phase.

In Phase #2, P-Coffee repeats the wall identification process for ×2 times to find more walls that are reproducible. Only walls that identified more than twice are accepted. In Table 3.1, these walls are denoted by "triple+", meaning the walls that appear "three times or more" in one phase. The acceptance qualification is set to triple+ in order to select more reliable walls in the earlier stages. Jumping is not allowed, because we expect that there still remain many undiscovered complete walls since only 5% of walls are fixed in the previous phase. Note that scores are not used for acceptance checking.

In Phase #3, lazy jumping is allowed so as to effectively identify walls that contain gaps. Complete walls may still remain. But, if this is the case, lazy jumping will not actually be used, since it is called only when there is no candidate child to be added at the current child depth. To find more reproducible walls, we raise the multiplier to ×4. Walls identified more than once are accepted. We denote these walls "double+" to represent the ones that appear "two times or more" in one phase. No scores are used for acceptance either in this phase.

Finally, in Phase #4, the multiplier ×2 is applied. In this phase, all non-conflicting walls are accepted. Priority is given to double+'s, and then, higher scorers. Phase #4 repeats itself until every wall is identified. The threshold is disabled in this phase, because we are trying to fix all the partition walls irrespective of the reliability.

3.5 Parameters

There are four parameters that have to be set before running P-Coffee. The threshold pair score is required for efficient running of the wall identification process. The multiplier and the acceptance qualification including the initial acceptance rate control the wall selection procedure. In this section, we will describe in detail the features that these parameters have.

3.5.1 The Threshold Pair Scores

As explained in Chapter 2, sequence identity (%) is the minimum score that is guaranteed to the aligned residue pairs belonging to the sequences. Admitting that sequence comparison is reliable down to the twilight zone (25~30% identity), we may conclude that we can ignore residue pairs with substitution score below this zone when extending a tree. But if multiple sequences are used in search of hidden residue equivalencies, interrelatedness among residues may be helpful in identifying reliable equivalencies below this limit. More importantly, it is impossible to construct an alignment solution with columns that are composed of only reliable residues, because in MSA we have to assign every residue to some position in the alignment, irrespective of its reliability. Therefore, in the big picture, we cannot ignore residues just because they entail substitution scores below the twilight zone.

Because of the lack of information sufficient to overcome this complexity, we decided the proper level of the threshold pair score experimentally by observing the effect of threshold values on algorithm performance.

If the threshold is set to a lower value, the branch factor will be larger, because many unreliable residues can be added to growing tree, and the algorithm will require a large

amount of computational resources. Note that this tree-based procedure has exponential time complexity, so controlling the branch factor is crucial. On the other hand, if we set the threshold excessively high, it will prevent the paths being constructed from reaching to the bottom depth without gaps, thus making them unlikely to survive the competition among paths in the tree. We want to find the optimum threshold value that controls the exponential time complexity without sacrificing the reliability of identified walls.

3.5.2 The Multiplier and The Acceptance Qualification

As described in Section 3.4.3, the *multiplier* specifies the number of iterations used to form a pool of partition walls in each phase, and the *acceptance qualification* describes a condition for the walls to be accepted. We use the maximum sequence length as a multiplicand so that each residue in the maximum length sequence can be examined probabilistically at least once. Note that the closest number to the resulting alignment length currently available is the maximum sequence length, because no two residues in the same sequence can be in the same alignment column.

The higher the value of the multiplier, the greater the chance that walls appear more than some specified number of times. But if the multiplier is set to a value beyond required, redundant computation will result. For example, the wall that has been identified twice in ×2 iterations is roughly expected to appear four times in ×4 iterations, if the residues contained in the wall are closely interrelated. Furthermore, even incorrect walls can pass this screen. This happens in case they contain a small portion of interrelated residues and similar hierarchies are used. On the other hand, if the multiplier is set to a value too small, the

number of qualified walls will be reduced, since walls do not have enough chance to appeal their reliability.

3.5.3 The Initial Acceptance Rate

The *initial acceptance rate* is one of the acceptance qualifications that is used only in the initial phase of P-Coffee. The motivation of using the initial acceptance rate is that there may exist a minimum score that we can use as a cutoff criterion when selecting a wall from the pool of identified walls. Since this minimum will be different case by case, we normalize it by the highest score found during the repetition of wall identification process.

The initial acceptance rate is defined by the following formula. So, for example, if the rate is set to 10%, we will accept only the top 10% highest scorers from the identified walls in Phase #1.

$$\begin{aligned} \textit{the initial acceptance rate} &= 1 - \textit{normalized minimum score for guaranteed hit} \\ &= 1 - \frac{\textit{the highest fail score}}{\textit{the overall highest score}} \end{aligned} \tag{3.1}$$

The initial acceptance rate is used only in Phase #1, since it is unlikely that we encounter such high scorers again in the later phases once we use up all high scorers. The number of accepted walls will be reduced as the acceptance rate gets smaller. So we want to estimate the maximum score level that we can trust without further analysis.

CHAPTER 4

Tests and Results

In the previous chapter, we explained some features that are introduced for more effective and accurate running of the algorithm. We pointed out four parameters to be determined before the running of P-Coffee. In this chapter, we will demonstrate how the parameters have been determined. And then, we will show the competency of P-Coffee by comparing its performance with T-Coffee using BAliBASE v1.0 test cases.

4.1 Tests

As clarified in Section 3.5, we can control the effectiveness of the P-Coffee by adjusting the parameters – the threshold pair score, the initial acceptance rate, the multiplier, and the acceptance qualification. In this section, we will suggest the ways to determine the optimum values of these parameters.

4.1.1 Test #1: The Optimum Threshold Pair Score

One of the most important benefits of BAliBASE is that we can evaluate the correctness of an identified wall by simply seeing whether it is really included in the reference alignment or not. We call it a *hit* if we can find a match, and a *fail* otherwise. Using this feature, we can obtain the hit rate of the identified walls after repeating the wall identification process for a specified number of times. In this test, we want to know to what extent the algorithm should allow unreliable (lower scored pair) residues to be added to a tree being constructed. So we can find out the optimum threshold value by selecting one that makes the maximum hit rate. We tested 5, 10, 15, and 20 as the threshold values with the multiplier value of ×1. We did not allow jumping in this test.

4.1.2 Test #2: The Optimum Initial Acceptance Rate

In this test, we wanted to find out the maximum level of acceptance rate that would cover most of the test cases. This test can be done alongside Test #1, since we can obtain the initial acceptance rate of each test case by simply keeping track of the highest fail score and the overall highest during the iteration.

4.1.3 Test #3: The Optimum Combination of the Multiplier and the Acceptance Qualification

As in Test #1, the optimum combination of the multiplier and the acceptance qualification can be obtained by seeing which combination achieves the maximum hit rate.

We tested the multiplier values of ×1, ×2, ×3, and ×4, and analyzed the hit rate as well as the number of identified walls for the qualification values of singles, double+'s, and triple+'s for each multiplier value. We used the threshold pair score of 10, and did not allow jumping in this test.

4.1.4 Test #4: The Performance of P-Coffee

BAliBASE also provides a program, BaliScore, so that we can score our results according to the standardized guideline. This score can be used to benchmark the performance of our algorithm with that of other MSA tools. Because it is reported that T-Coffee outperforms other well-known MSA tools, we benchmarked only with T-Coffee in this test.

4.2 Test Results

4.2.1 The Optimum Threshold Pair Score

As expected, if the threshold pair score is smaller, an exponentially larger amount of CPU time is used to identify partition walls, because the branch factor of trees will increase. The result is shown in Table 4.1 and Figure 4.1. But lower thresholds did not always result in higher hit rates. This means that a branchy tree does not guarantee finding more reliable partition walls. Rather, in many cases, hit rates decreased with the threshold values lower than the optimums, because it is more likely that unnecessary residues are included in the surviving walls.

Table 4.1: The performance of the partition wall identification with the varying threshold pair scores. 1idy (ref 2), 1lvl (ref2), 1tgxA (ref2), and 1idy (ref3) are excluded from this result to avoid a bias that can be arise as a result of their excessive amount of CPU time (in seconds) used.

	Threshold = 5		**Threshold = 10**		Threshold = 15		Threshold = 20	
	Hit Rate	CPU Time	**Hit Rate**	**CPU Time**	Hit Rate	CPU Time	Hit Rate	CPU Time
Ref 1	0.819	47	**0.828**	**53**	0.793	35	0.747	42
Ref 2	0.542	15200	**0.542**	**5043**	0.400	1141	0.305	811
Ref 3	0.541	18242	**0.565**	**1265**	0.423	227	0.317	157
Ref 4	0.575	425	**0.519**	**211**	0.469	92	0.383	88
Ref 5	0.730	576	**0.731**	**216**	0.726	96	0.692	95
Overall	0.714	3623	**0.707**	**981**	0.651	239	0.588	184

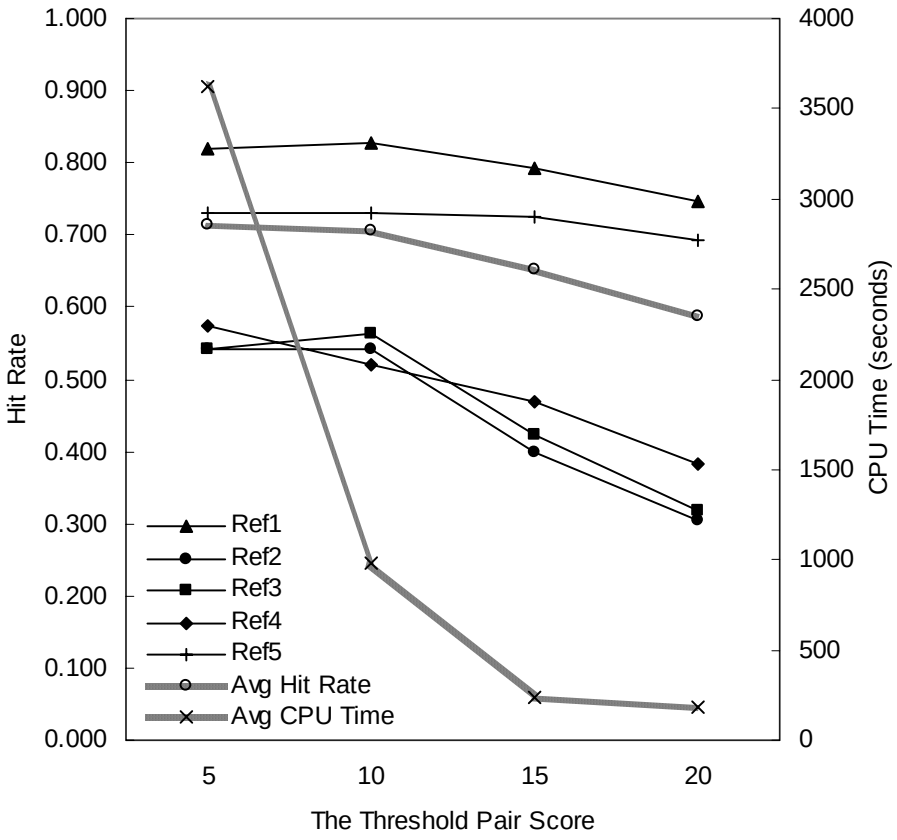


Figure 4.1: Effect of the threshold pair score on the performance of the wall identification process. The average hit rate declines from the threshold value of 10.

Data shows that the most suitable threshold pair score is 10, since, on average, the hit rate keeps its level even though it pays only 981 / 3623 ≈ 27% of CPU time compared with the case that the threshold of 5 is used. The optimum threshold value proves that we cannot ignore a residue just because it has a substitution score below the twilight zone sequence identity.

4.2.2 The Optimum Initial Acceptance Rate

For each threshold value that was used in the previous test, the initial acceptance rates were observed. The result is summarized in Table 4.2. For example, the cell that is marked (*) in Table 4.2 means that only the top 3.9% of the highest scoring walls were correct in Reference 2 test cases when we use the threshold value of 10. Walls with scores below that level are no longer safe to use to construct alignment solutions without further analysis.

The rates turn out to be insensitive to the threshold values. This is because the highest fail score and the overall highest could be found no matter what threshold value below the twilight zone is used.

Table 4.2: Reliable score range of sum-of-pairs scoring scheme

Threshold	5	**10**	15	20
Ref 1	0.209	**0.207**	0.210	0.200
Ref 2	0.039	**(*) 0.039**	0.039	0.041
Ref 3	0.071	**0.070**	0.063	0.066
Ref 4	0.092	**0.086**	0.087	0.089
Ref 5	0.199	**0.137**	0.140	0.135
Overall	0.156	**0.149**	0.151	0.145

The data shows that, on average, only the top 15% highest scorers were reliable under the sum-of-pairs scoring scheme. But, considering the narrowest range among all categories (see the cell marked (*) in Table 4.2), we decided to accept only the top 5% highest scorers in the first phase of P-Coffee. Figure 4.2 illustrates how many categories the optimum initial acceptance rate covers.

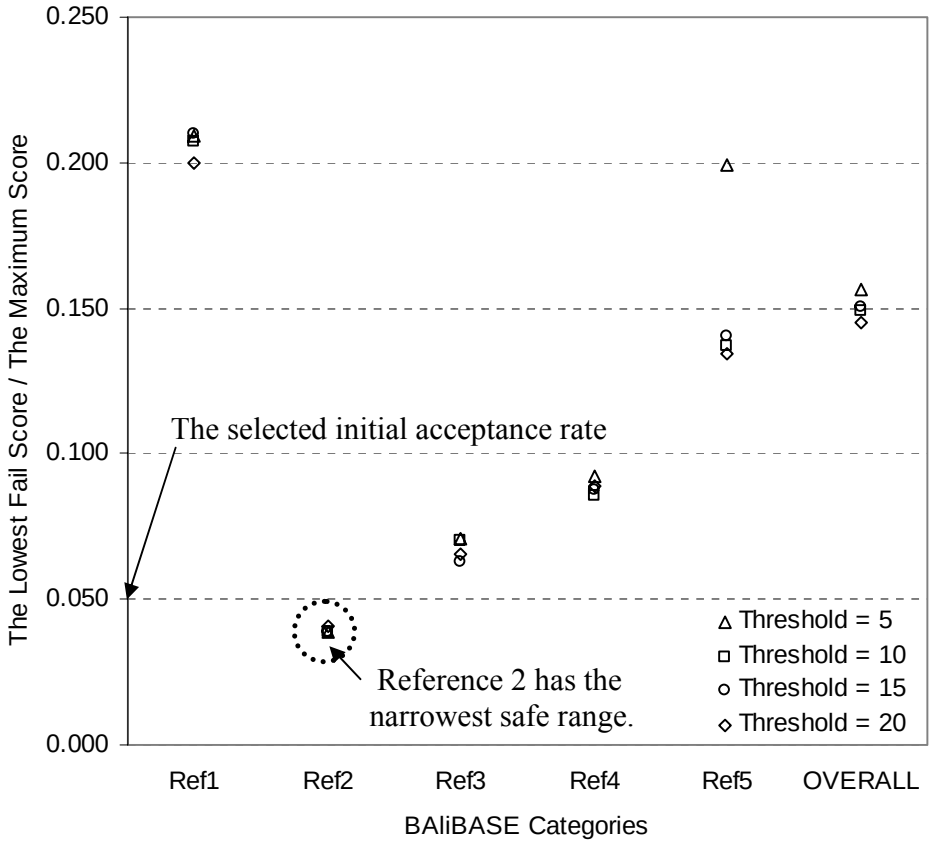


Figure 4.2: The reliable range of the sum-of-pair scores

The selected initial acceptance rate does not fully cover the test cases in Reference 2. But even incorrect walls with scores near this region will not harm the reliability of an overall alignment solution, since they would not have reached such high scores without containing enough of the residue equivalencies we are looking for.

4.2.3 The Optimum Combination of Multiplier-Qualification

Various combinations of multiplier-qualification values were tested. As expected, the accuracy grew in the order of singles, double+'s, and triple+'s throughout the categories. As mentioned in Section 3.5.2, the accuracy declined as the multiplier was set to higher values. From the data provided in Table 4.3, we have decided the values of the multiplier and the acceptance qualification to be ×2-triple+. Compared with the most accurate combination, ×1-triple+ (see underlined cells in Table 4.3), ×2-triple+ identified (9767 – 2732) / 2732 ≈ 258% more partition walls without loss of the hit rate. The combination ×3-triple+ (see double underlined cells) found even more partition walls than the optimum combination without significant loss of accuracy, but it had some weakness for Reference 4 test cases (see the cell marked (*)). On the other hand, the selected combination of ×2-triple+, was robust to the type changes of the problem set. This is also illustrated in Figure 4.3.

Table 4.3: The reliability of reproducible walls in various combinations of multiplier-qualification. The test cases, 1idy, 1lvl, and 1tgxA (all in reference 2), are excluded from this result. Because we counted the walls that are identified more than three times as three, the raw hit rates are estimated slightly lower than the real values.

Qualification	Singles			
Multipliers	×1	×2	×3	×4
Ref1	0.681	0.479	0.304	0.176
Ref2	0.338	0.186	0.100	0.057
Ref3	0.345	0.177	0.090	0.050
Ref4	0.232	0.154	0.169	0.111
Ref5	0.510	0.275	0.156	0.097
Overall	0.497	0.298	0.190	0.110
Num of Walls	11272	9137	6861	4783
Qualification	Double+			
Multipliers	×1	×2	×3	×4
Ref1	0.945	0.928	0.905	0.879
Ref2	0.895	0.856	0.844	0.830
Ref3	0.988	0.979	0.968	0.961
Ref4	0.826	0.757	0.660	0.593
Ref5	0.948	0.942	0.923	0.903
Overall	0.928	0.907	0.878	0.848
Num of Walls	8115	17473	23186	26545
Qualification	Triple+			
Multipliers	×1	×**2**	×3	×4
Ref1	0.966	**0.961**	0.961	0.950
Ref2	0.919	**0.911**	0.901	0.894
Ref3	0.987	**0.993**	0.987	0.984
Ref4	0.925	**0.908**	(*) 0.841	0.802
Ref5	0.959	**0.978**	0.965	0.962
Overall	0.954	**0.953**	0.946	0.934
Num of Walls	2732	**9767**	16794	21840
	Raw (estimated)			
Multipliers	×1	×2	×3	×4
Overall	0.676	0.602	0.520	0.446

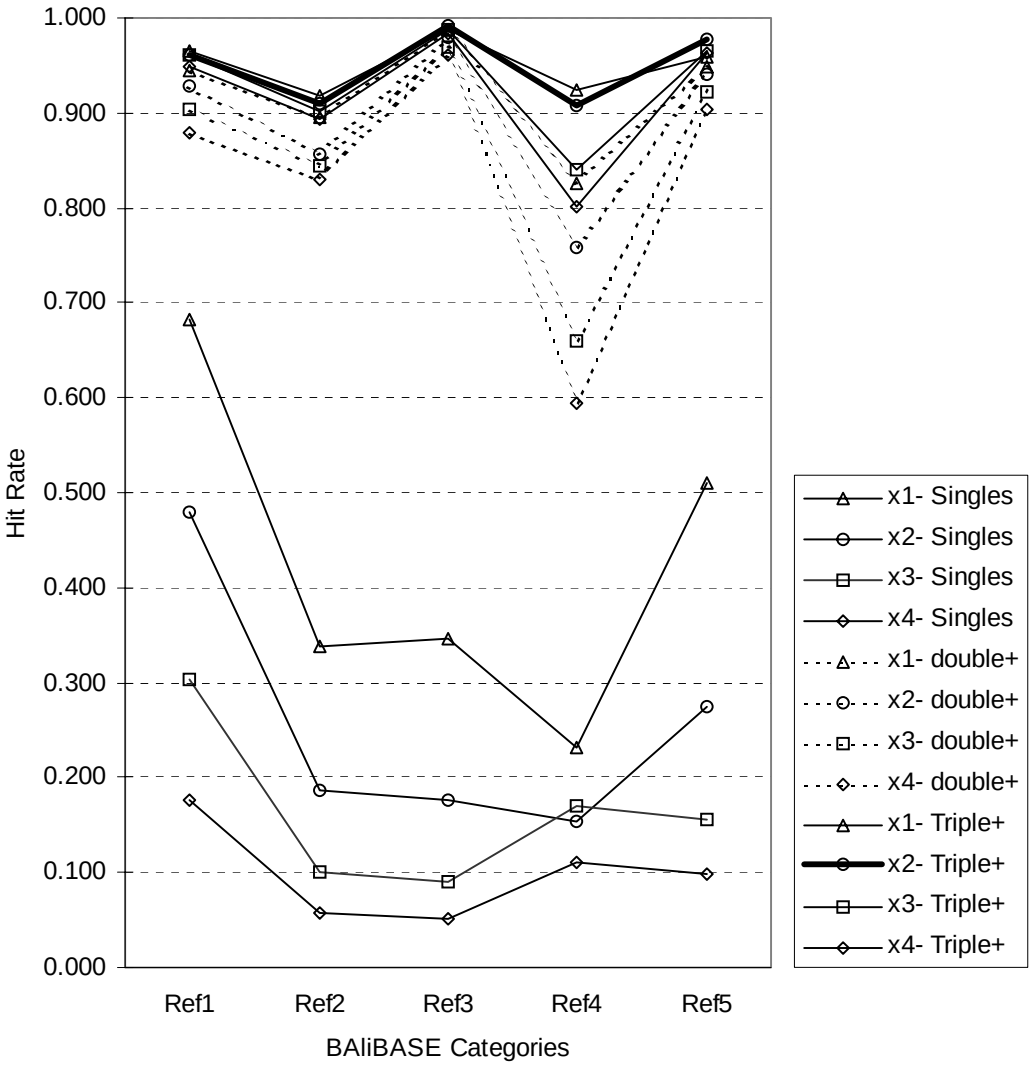


Figure 4.3: The surge in accuracy when we use reproducibility criterion

The overall average accuracy of each qualification is illustrated in Figure 4.4. Compared with the raw accuracy, the selected combination achieved an accuracy increase of (0.953 – 0.602) / 0.602 ≈ 58%. Note that the limit we could have achieve was (1 – 0.602) / 0.602 ≈ 66%. Since over 95% of walls identified in Phase #2 are hits, this will significantly contribute to the reliability of overall alignment solution.

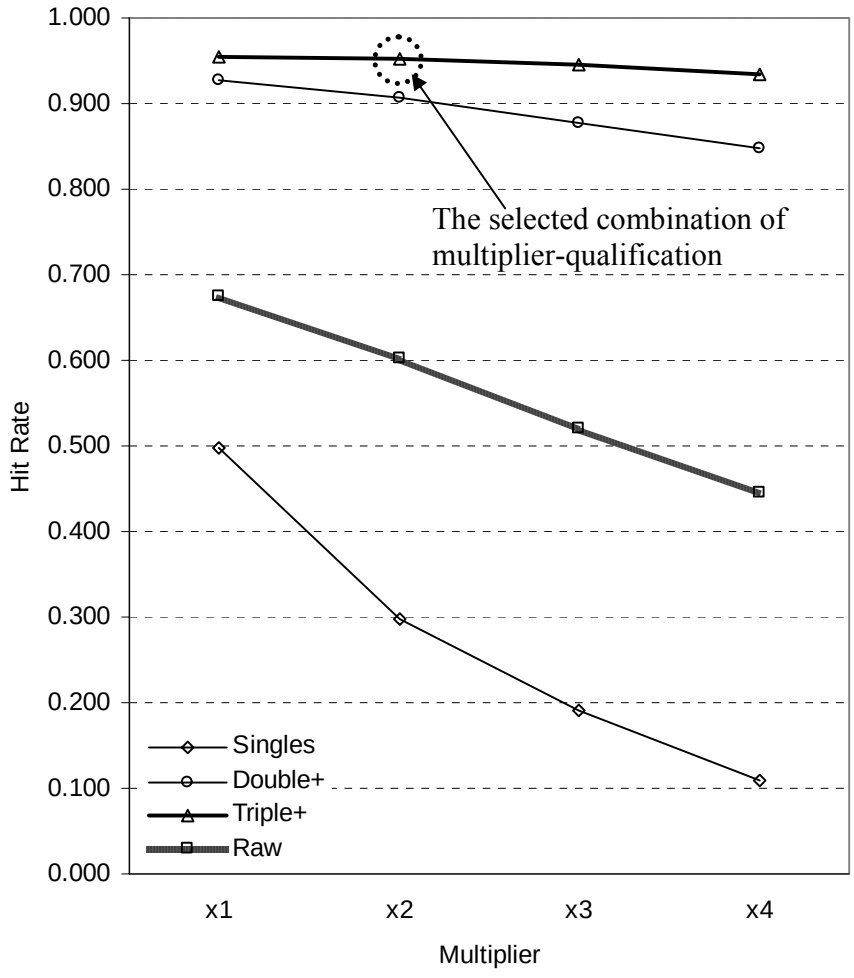


Figure 4.4: Overall accuracy of each acceptance qualification

The numbers of correct walls that are identified during the tests are illustrated in Figure 4.5. As mentioned in Section 3.5.2, the number of singles decreased as the multiplier was set to a larger value. Instead, more double+'s and triple+'s could be identified.

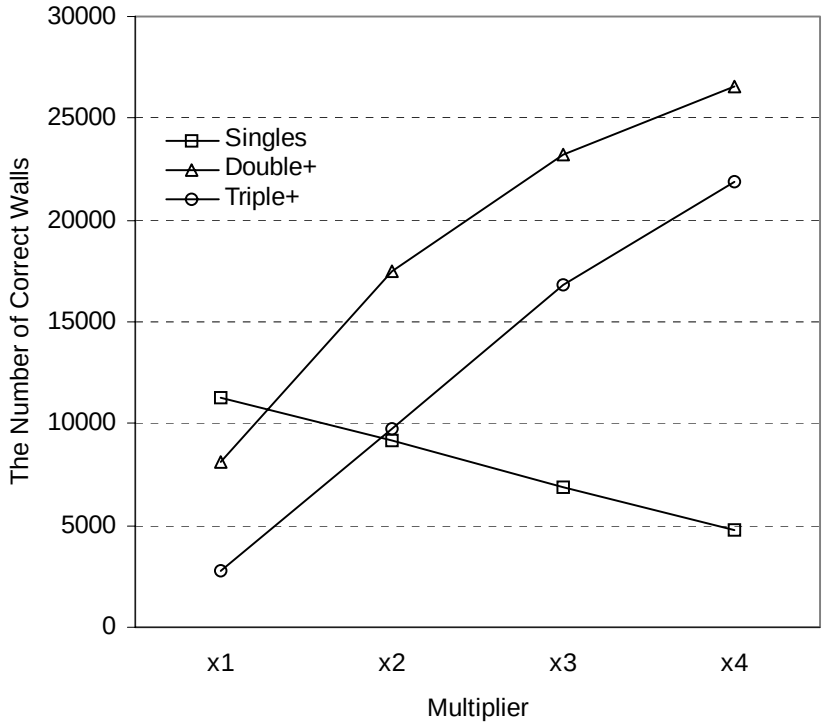


Figure 4.5: The number of correct walls identified

Using the selected combination of ×2-triple+, we were able to identify total 9767 correct partition walls. Since the total alignment length of all the test sequence set is 49925, the number of identified walls corresponds to 20% of all the columns to be found. This may

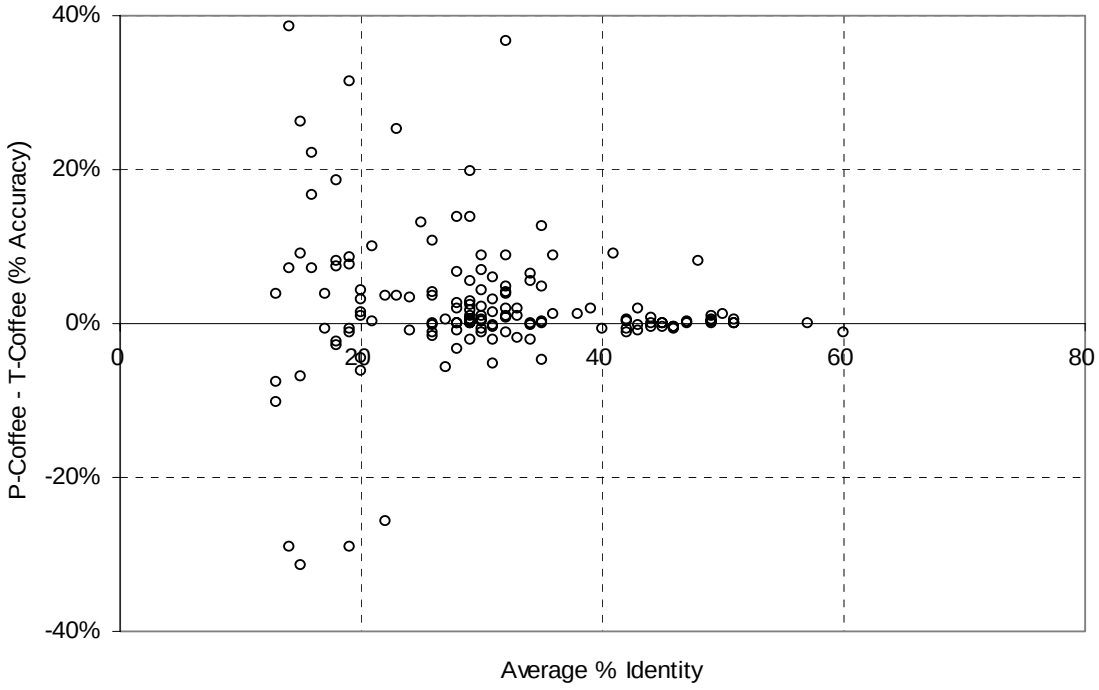
look like a very small portion, but with this amount of walls, the average partition size is reduced to less than 100% / 20% = 5. Since residues outside a partition are no longer considered when a tree is being built in the partition, the probability that unreliable residues are attached to the tree also decreases. This is why we apply relatively loose qualification condition (which is ×4-double+) to Phase #3. We have not tested for this combination, because we can indirectly show its effectiveness with the accuracies of the final alignment solutions.

4.2.4 The Performance of P-Coffee

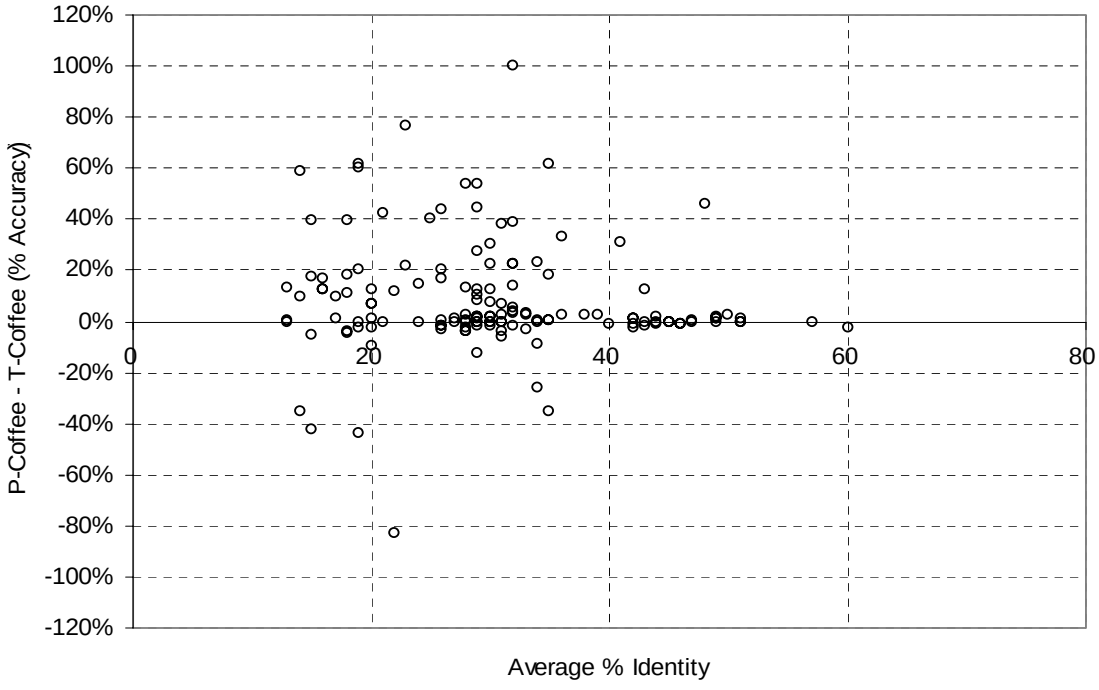
The competency of P-Coffee is demonstrated by comparing its performance with that of T-Coffee. The differences between the percent accuracy achieved by P-Coffee and T-Coffee are plotted along with the average sequence identity of test sets. Both the residue pair accuracy and the column accuracy are compared, and each result is drawn separately as in Figure 4.6 (a) and (b). To make the differences clearer, we divided the data into two parts, Reference 1 and the others. As shown in Figure 4.7, we could not identify any improvement for the test cases in Reference 1. But, for References 2 to 5, which are composed of problem sets with more sequences, P-Coffee improved both the residue pair accuracy and the column accuracy on average by 6% and 18% respectively. As validated by the P-values for the Wilcoxon signed rank tests both less than 10^{-7}, the difference between P-Coffee and T-Coffee is significant. This improvement can also be easily noticed by the dispersion of data points shifted towards upper region of x-axis in Figure 4.8.

Table 4.4: The average accuracy P-Coffee compared with that of T-Coffee.

	SP Score		TC Score	
	T-Coffee	P-Coffee	T-Coffee	P-Coffee
Ref 1	94.1%	93.8%	89.3%	89.9%
Ref 2	86.6%	93.1%	35.5%	59.6%
Ref 3	92.2%	97.5%	62.9%	85.0%
Ref 4	78.5%	85.5%	57.0%	70.2%
Ref 5	93.7%	97.5%	82.8%	90.8%
Overall	91.1%	93.5%	74.6%	82.7%
Ref 2~5	**87.1%**	**92.9%**	**55.2%**	**73.1%**

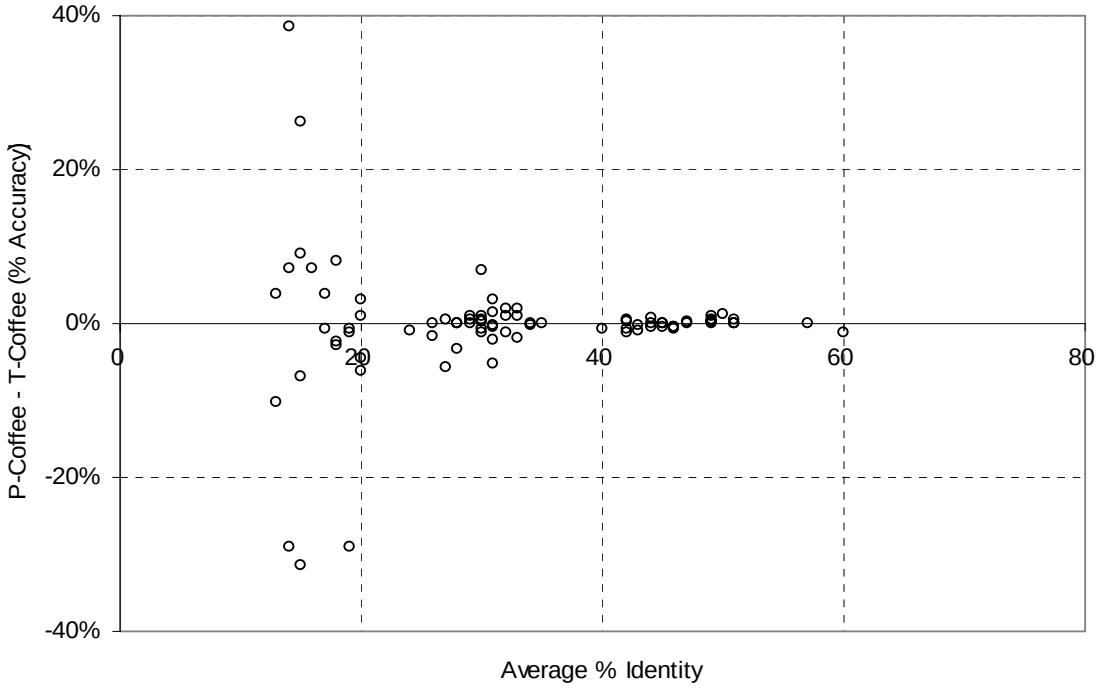


(a)

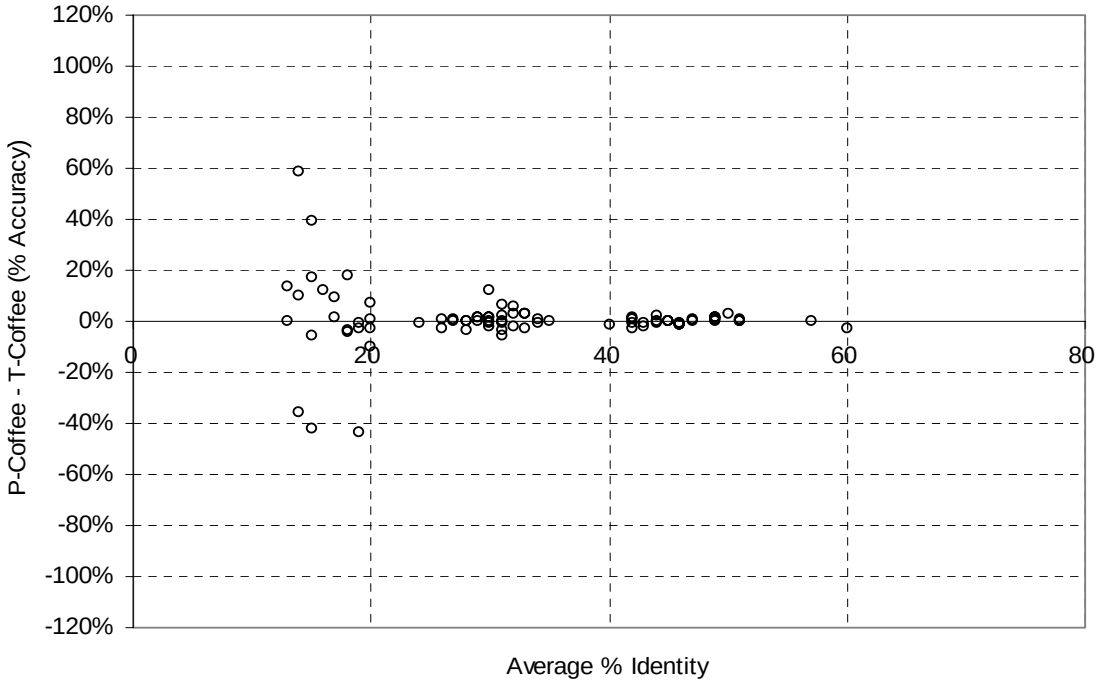


(b)

Figure 4.6: The difference in (a) SP Score and (b) TP Score (Overall)

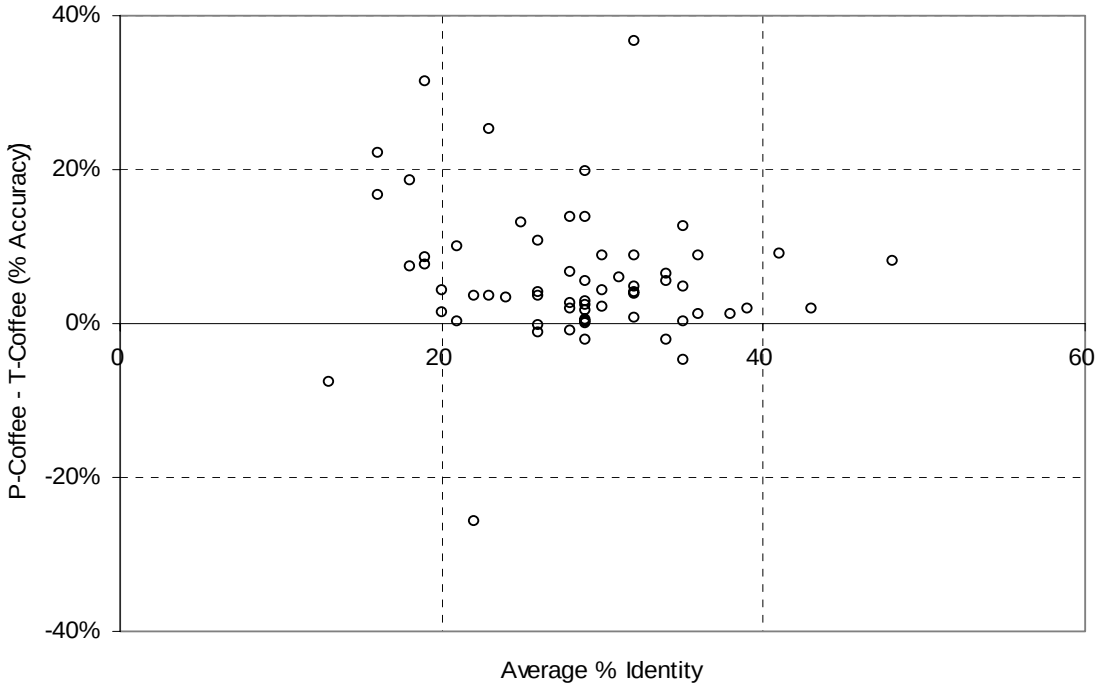


(a)

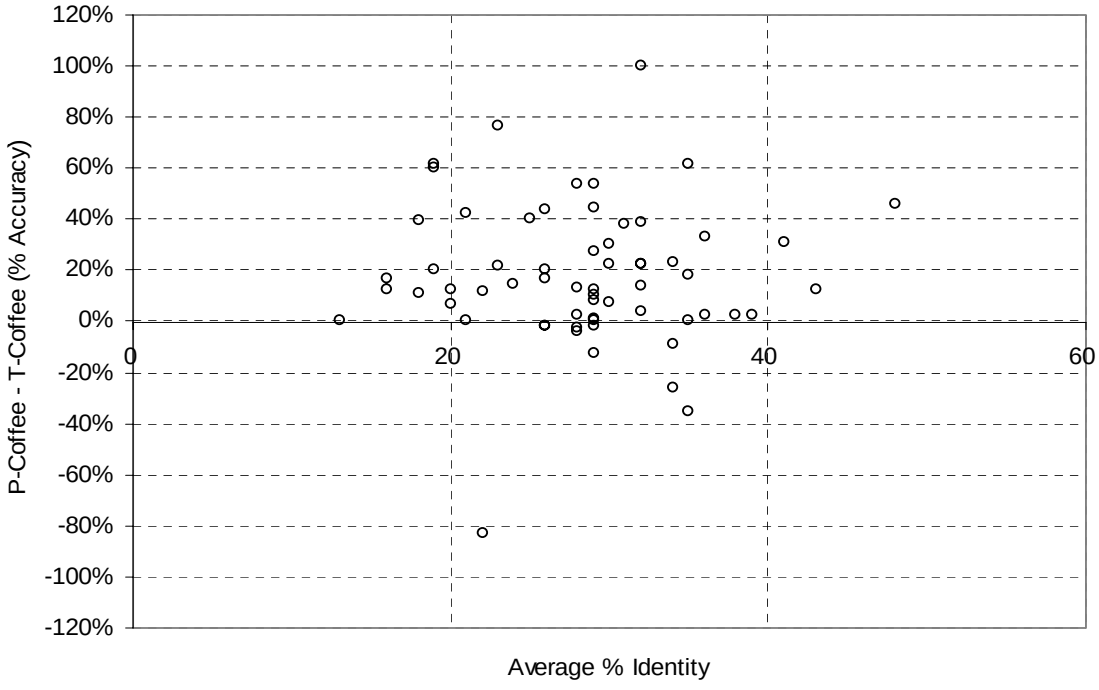


(b)

Figure 4.7: The difference in (a) SP Score and (b) TP Score (Ref1 Only)



(a)



(b)

Figure 4.8: The difference in (a) SP Score and (b) TP Score (Ref2~5)

4.2.5 Further Analysis of the Performance of P-Coffee

P-Coffee outperformed T-Coffee in References 2 to 5 test cases. These categories contain 15.5 sequences on average, while Reference 1 contains only 4.5 sequences. If the number of sequences increases, the length of partition walls also increases. As a result, the probability that a certain wall is identified more than once gets smaller, since more residues should be interrelated. This makes the reproducible walls more reliable than those in cases with fewer number of sequences. As shown in Figure 4.9, the hit rates of both double+'s and triple+'s are distributed in the higher value range as the number of sequences increases.

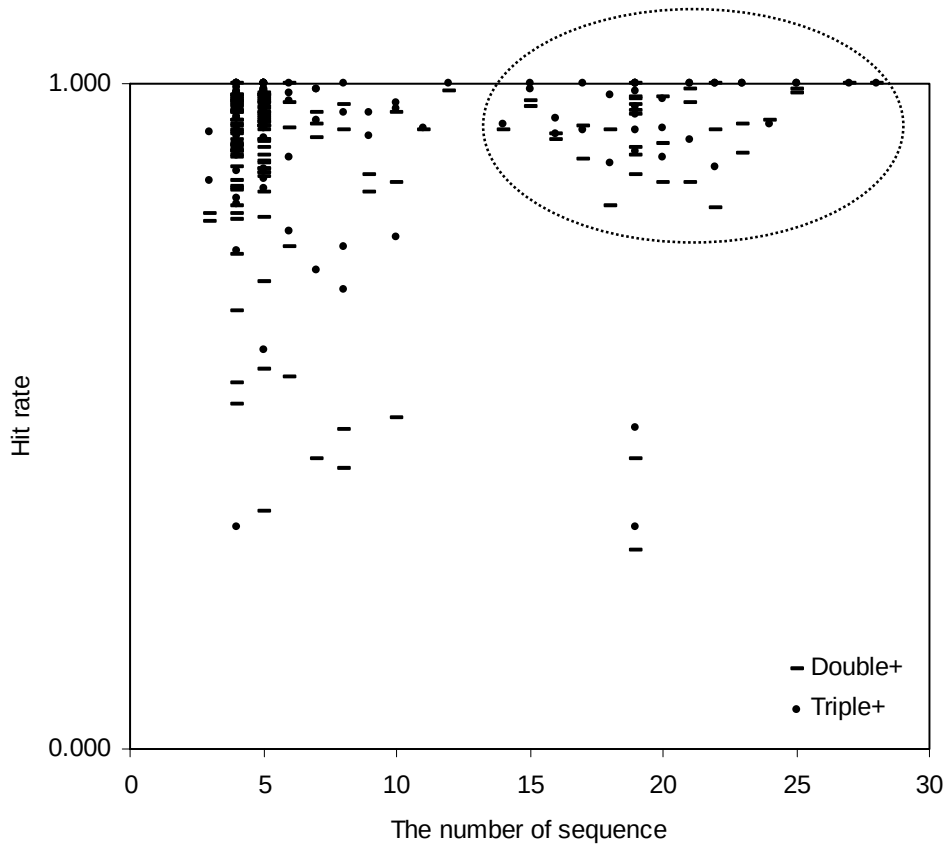


Figure 4.9: The relationship between the number of sequences and the accuracy of reproduced walls

4.2.6 The Run Time Analysis of P-Coffee algorithm

We analyzed the run time of P-Coffee with a fixed number of sequences, in order to prove the advantage of partitioning. If the number of sequences is fixed, the run time of the algorithm is expected to be O(L), where L is the average length of the sequences, because the number of partition walls to be identified will increase proportional to the average length of the sequence set in most cases. The analysis was done only with Reference 1 test cases, because it is the only category that has enough data points that share the same number of sequences. As illustrated in Figure 4.10 and Figure 4.11, the CPU time increased linearly in the length of the sequence set.

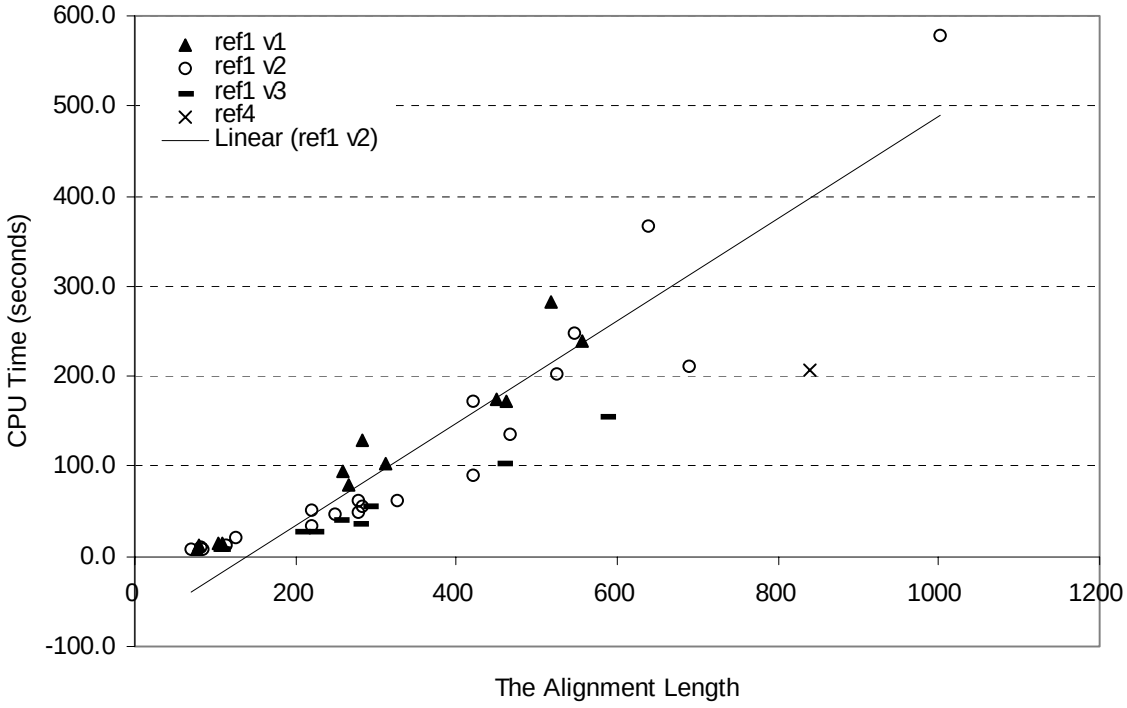


Figure 4.10: Run time analysis of P-Coffee with a fixed number of sequences (4-sequence cases).

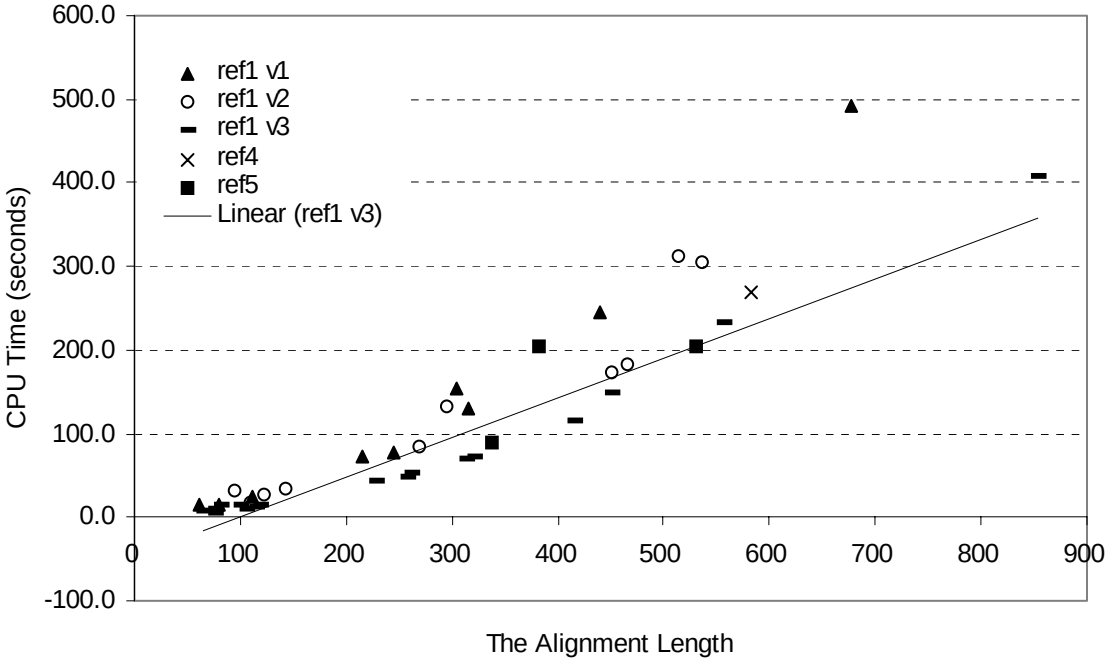


Figure 4.11: Run time analysis of P-Coffee with a fixed number of sequences (5-sequence cases). The trend line in each graph of Figure 4.10 and Figure 4.11 is drawn only for the sub-category that has the maximum range in the sequence length. We used the length of the reference alignment instead of the average sequence length in this graph.

We also examined the run time with varying numbers of sequences to observe the influence of the tricks we have introduced in order to speed up the algorithm. Note that this test was not possible for Reference 1 test cases, since they are composed of the sequence sets with 4 to 6 sequences. As shown in Figure 4.12, the tricks worked to hold the run time within roughly the linear time range for reference 4 and 5 test cases, except for one outlier. This proves that the techniques used for speed up are properly handling the exponential time complexity at least for BAliBASE test cases. But, we do not believe that the tricks can

guarantee to hold an arbitrary sequence sets within the linear time complexity, as suggested by the outlier.

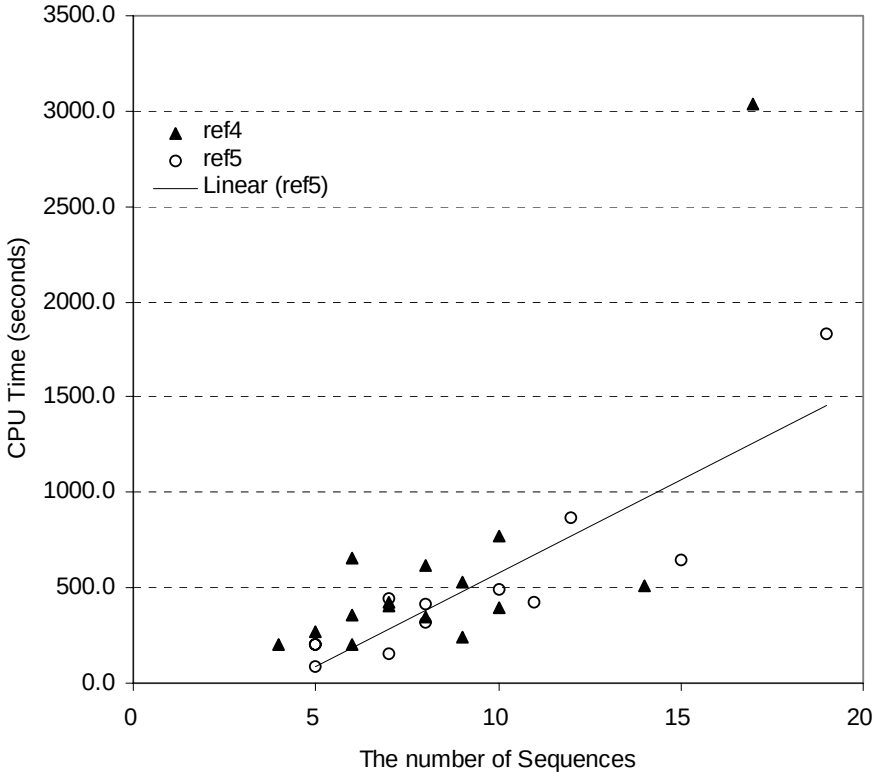


Figure 4.12: Run time analysis with varying number of sequences (Ref 4 and 5 only). A trend line is added for reference 5 cases.

There are various environmental factors that can affect the time complexity of P-Coffee: for example, average sequence identity, the distribution of residue pair scores, the compositional features of sequence set (as in BAliBASE categories), or the size of the T-Coffee library. If a problem set contains a large number of sequences, the effect of such

environmental factors can be augmented. For this reason, it was difficult to analyze the influence of the techniques for speedup. This is well illustrated in Figure 4.13 by the widespread dispersion of data points. That is, for References 2 and 3 categories, it was not possible to identify any trend in the run time with varying number of sequences.

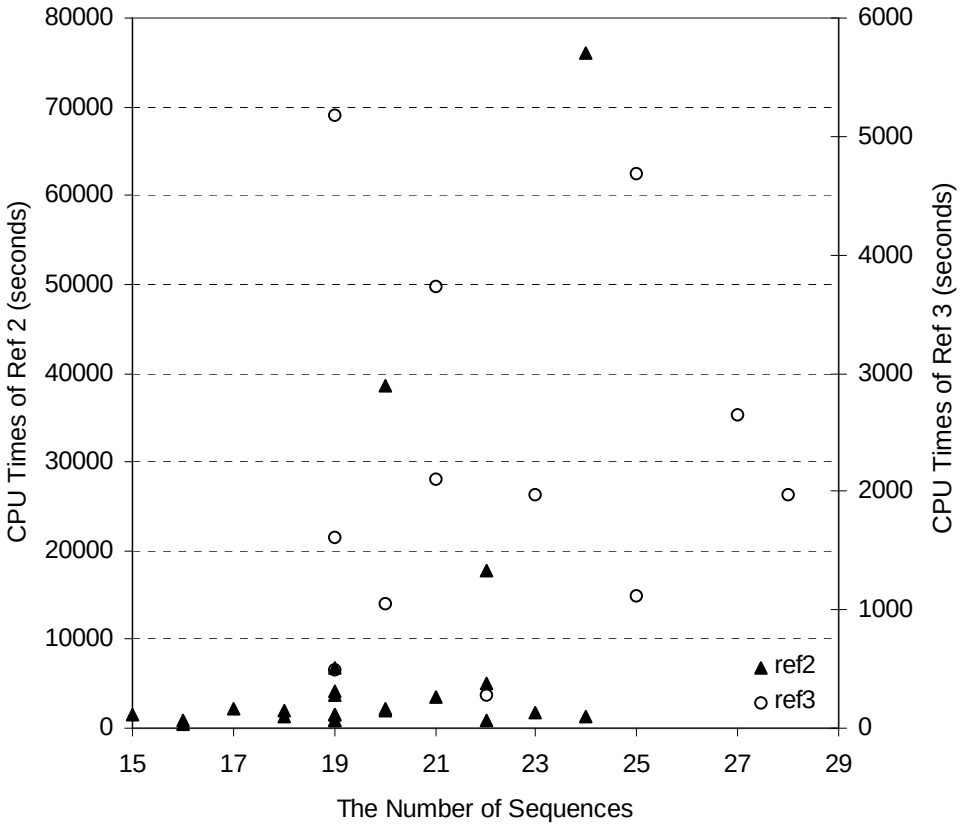


Figure 4.13: Run time analysis with varying number of sequences (Ref 2 and 3 only)

CHAPTER 5

Conclusion and Future Work

In this research, we propose a new divide-and-conquer method, P-Coffee, for the problem of multiple sequence alignment. The main differences from other MSA methods is that the new method aligns sequences column by column, and that the method does not try to maximize (or minimize) any objective function to reach a solution. In addition, P-Coffee deals with all the sequences in a set simultaneously. Although P-Coffee used 25.7 times more CPU time than T-Coffee, it outperformed T-Coffee by 5.9% in residue pair accuracy, and 18.0% in column accuracy for References 2 to 5 categories. For test cases in Reference 1, the performance was almost same. P-Coffee has a relatively better performance on problem sets that contain a larger number of sequences. In addition, P-Coffee was robust to the compositional features of sequence sets such as sets containing orphans, sets with divergent families, sets with N/C-terminal extensions, and sets with internal insertion.

P-Coffee has two core techniques invented in this research. One is the tree-based identification process of an alignment column (our term for this is “partition wall”). In this process, the algorithm builds a tree according to a randomly generated sequence hierarchy, using the substitution scores in the T-Coffee extended library (a position-specific scoring scheme). And then, it identifies one partition wall by taking the path with the highest sum-of-pairs score.

The other innovative technique is the test method that distinguishes probabilistically more reliable walls out of the pool of identified walls. We invented this reliable test method based on the idea that the walls containing closely interrelated residues can be identified again with trees using the different hierarchies. In this test method, by accepting the walls that identified more than a specified number of times, we could significantly improve the accuracy of selected walls. The reproducible walls turned out to be even more reliable if the number of sequences in the set gets larger.

The most significant achievement of this research is that the wall identification and selection process, two major components of P-Coffee, can be used in any MSA tool as a pre-processor that reduces the dimension of the problem by dividing the sequence set into smaller partitions. This process is a potential breakthrough for most of iterative alignment tools that have poor performance in large sequence sets. In fact, another divide-and-conquer algorithm, DCA (divide and conquer alignment) designed by Stoye et al [30], improved the usability of an exact alignment method (the MSA program [31]).

DCA algorithm computes a potential *cut position* using a heuristic function, and divide a sequence set until the size of subsets reduces small enough so that the MSA program can be applied. On the other hand, P-Coffee has its uniqueness in that it does not depend

upon other alignment methods. This is possible since a sequence set is divided into partitions using identified columns that correspond to a part of an alignment solution. The advantage of our approach is that we minimized the sources of bias by not requiring any heuristic function to optimize, phylogenetic tree, nor gap cost scheme (the parameters frequently used in other sequence alignment methods for biologically proper distribution of gaps in an alignment solution).

There is still a room for improvement in P-Coffee. First, we may be able to save time by partitioning the T-Coffee library as well, whenever new sub-partitions are obtained. Since the T-Coffee library contains all the position-specific substitution scores of residue pairs, the data structure becomes large to hold the entire library. It is inefficient to scan through the entire library whenever the algorithm looks up the required residue pairs. Considering that the library scan happens whenever a node is looking for its candidate child nodes, partitioning the library can save a large amount of look up time. Second, we may be able to reduce the total number of wall identification activities by keeping the walls with a time stamp, instead of discarding all of them after wall selection is completed in each phase. Some unfairness can be brought about by this change, because walls that are found in an earlier phase would have an unwanted advantage in meeting the acceptance qualification. This problem may be resolved by giving a penalty to those walls in the form of an exponential decay function. Finally, the performance of P-Coffee can be improved by incorporating more reliable information additionally into the T-Coffee library. In fact, research related to this matter is already reported. According to O'Sullivan et al, they integrated the results of pairwise structure alignment tools such as SAP [32] and LSQman [33] into the primary library of T-Coffee with weight of 100 [29]. This kind of efforts will amplify the difference

between reliable residue pairs and others, and therefore the sum-of-pairs score will become more reliable. Then, of course, we may be able to raise the initial acceptance rate.

Bibliography

[1] M.B. Swindells, C.A. Orengo, D.T. Jones, E.G. Hutchinson, and J.M. Thornton. Contemporary approaches to protein structure classification. *BioEssays*, **20**:884-891, 1998.

[2] J.U. Bowie, R. Lüthy, and D. Eisenberg. A method to identify protein sequences that fold into a known three-dimensional structure. *Science*, **253**:164-170, 1991.

[3] D.T. Jones, W.R. Taylor, and J.M. Thornton. A new approach to protein fold recognition. *Nature*, **358**:86-89, 1992.

[4] C. Crothia, and A.M. Lesk. The relation between the divergence of sequence and structure in proteins. *The EMBO Journal*, **5**:823-826, 1986.

[5] R.F. Doolittle. *Of URFs and ORFs: a primer on how to analyze derived amino acid sequences*. University Science Books, Mill Valley, CA, USA, 1986.

[6] C. Notredame. Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics*, **3**(1):131-144, 2002.

[7] L. Wang, and T. Jiang. On the complexity of multiple sequence alignment. Journal of Computational Biology, **1**:337-348, 1994.

[8] J.D. Thompson, F. Plewniak, and O. Poch. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, **15**:87-88, 1999.

[9] S.W. Perrey, J. Stoye, V. Moulton, and A.W.M. Dress. On Simultaneous versus Iterative Multiple Sequence Alignment. Submitted, 1997. http://citeseer.ist.psu.edu/perrey97simultaneous.html

[10] S.B. Needleman, and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology,* **48:**443-453, 1970.

[11] J. Devereux, P. Haeberli, and O. Smithies. GCG package. Nucleic Acids Research, **12**:387-395, 1984.

[12] F. Corper. Multiple sequence alignment with hierarchical clustering. *Nucleic Acids Research*, **16**:10881-10890, 1988.

[13] J.D. Thompson, D.G. Higgins, T.J. Gibson. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, **22**:4673-4690, 1994.

[14] W.R. Taylor, A flexible methods to align large numbers of biological sequences. *Journal of Molecular Evolution*, **28:**161-169, 1988.

[15] R. D. Smith, and T. F. Smith. Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative modeling. *Protein Engineering, Design and Selection*, **5**(1):35-41, 1992.

[16] C. Notredame, D.G. Higgins, and J. Heringa. T-Coffee: a novel method for multiple sequence alignments. *Journal of Molecular Biology*, **302**:205-217, 2000.

[17] S.R. Eddy. Multiple alignment using hidden Markov models. *Intelligent Systems for Molecular Biology*, **3**:114-120, 1995.

[18] C. Notredame, and D.G. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, **24**:1515-1524, 1996.

[19] T. Riaz, Y. Wang, and K. Li. Multiple sequence alignment using tabu search. *Conferences in Research and Practice in Information Technology*, **29**:223-232, 2004.

[20] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, and D. Haussler. Hidden Markov Models in Computational Biology: Applications to Protein Modeling. *Journal of Molecular Biology*, **235**(5):1501-1531, 1994.

[21] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, and J.C.Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**:208-214, 1993.

[22] O. Gotoh. Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinements as Assessed by Reference to Structural Alignments. *Journal of Molecular Biology*, **264**(4):823-838, 1996.

[23] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure* (M.O. Dayhoff ed.), **5**(3):345-352, National Biomedical Research Foundation, Silver Spring, Washington D.C., 1978.

[24] S. Henicoff, and J.G. Henicoff. Amino acid substitutuion matrices from protein blocks. *Proceedings of the National Academy of Sciences USA*, **89**:10915-10919, 1992.

[25] C.A. Orengo, D.T. Jones, and J.M. Thornton. *Bioinformatics: Genes, Proteins, & Computers*. Bios Scientific Publishers Ltd., Oxford OX4 1RE, U.K., 2003.

[26] C. Notredame, L. Holm, and D.G. Higgins. COFFEE: an objective function for multiple sequence alignments, *Bioinformatics*, **14**(5):407-422, 1998.

[27] X. Huang, and W. Miller. A time-efficient linear-space local similarity algorithm. *Advances in Applied Mathematics*, **12**:337-357, 1991.

[28] J.D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, **27**(13):2682-2690, 1999.

[29] O. O'Sullivan, K. Suhre, C. Abergel, D.G. Higgins, and C. Notredame. 3DCOFFEE: Combining Protein Sequences and Structures within Multiple Sequence Alignments. *Journal of Molecular Biology*, **340**:385-395, 2004.

[30] J. Stoye, V. Moulton, A.W. Dress. DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. Computer Applications in the Biosciences, **13**(6):625-626, 1997.

[31] D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences USA*, **86**:4412-4415, 1989

[32] W.R. Taylor, and C.A. Orengo. Protein structure alignment. *Journal of Molecular Biology*, **208**:1-22, 1989.

[33] G.J. Kleywegt. Use of non-crystallographic symmetry in protein structure refinement. *Acta Crystallographica D*, **52**:842-857, 1996.