High performance computing on biological sequence aligment

Miquel Orobitg Cortada

Dipòsit Legal: L.290-2013
http://hdl.handle.net/10803/110930

Escola Politècnica Superior
Departament d'Informàtica i Enginyeria Industrial

High performance computing on biological sequence alignment

Memòria presentada per obtenir el grau de
Doctor per la Universitat de Lleida
per

Miquel Orobitg Cortada

Dirigida per

Dr. Fernando Cores Prado **Dr. Fernando Guirado Fernández**
Universitat de Lleida

Programa de doctorat en Enginyeria

Lleida, febrer de 2013

Abstract

Multiple Sequence Alignment (MSA) is an extremely powerful tool for such important biological applications, as phylogenetic analysis, identification of conserved motifs and domains and structure prediction. The main idea behind MSA is to place protein residues into columns according to selected criteria. These criteria can be the structural, evolutionary, functional or sequence similarity. The first three criteria are based on biological meaning, but the fourth is not. Then the best match alignment may not exhibit the best biological meaning.

The other issue is that MSAs are computationally difficult to calculate, and most formulations of the problem lead to NP-hard optimization problems. MSA computation therefore depends on approximate heuristics or algorithms. Accurate and fast construction of MSAs has been extensively researched in recent years, and a variety of methods have been developed.

Nowadays, the new challenges in the genomics era are focused on aligning thousands, or even hundreds of thousands, of sequences. For this reason, alignment methods must be adapted to avoid being relegated from everyday use. Many MSA methods have introduced High Performance Computing capabilities to take advantage of the new technologies and infrastructures. In particular, there are many approaches that improve alignment performance by parallelizing the whole MSA application. However, these distributed methods exhibit scalability problems when the number of sequences increases, as they are constrained by data dependencies that guide the alignment process.

In this thesis, three different approaches to solving or reducing some limitations of the MSA methods are proposed. The first of these is aimed at reducing data dependencies to increase the degree of parallelism in the applications and

improve their scalability. The second one is devoted to reducing the memory requirements of a consistency-based method. And the last one is designed to increase the alignment accuracy by improving the guide tree.

From the parallelism point of view, we propose a new guide tree construction algorithm to exploit the degree of parallelism in the final alignment process in order to resolve the bottleneck of the progressive alignment stage in MSA. This new contribution increases the number of parallel alignments to take advantage of increasing computer resources by balancing the guide tree, but taking biological accuracy properties into account. The results achieved demonstrated that the proposed heuristic is capable of increasing the degree of parallelism in the parallel MSA method, thus reducing the execution time while maintaining the biological quality of the alignment.

Regarding the memory requirements, we propose a new approach to reduce the computational requirements of the consistency-based method, T-Coffee. The main goal of the proposal is to reduce the number of consistency data stored. This allows a reduction in the execution time and therefore an improvement in the scalability of TC to enable the method to align more sequences. Furthermore, the proposed methodology is focused on achieve a minimal the impact over accuracy of the alignments and it is the user who decides the degree of optimization. The results obtained proved that this approach is able to reduce the memory consumption and increase both performance and the number and the length of the sequences that the method can align.

In connection with biological accuracy, we propose Multiple Trees Alignment (MTA). MTA is a new MSA method for aligning multiple guide-tree variations for the same input sequences, evaluating the alignments obtained and selecting the best one as a result. MTA is implemented in parallel to provide a good compromise between time and accuracy. Besides, this approach could be applied to any progressive alignment method that accepts a guide tree as an input parameter, and the resulting alignments could be evaluated with any scoring function. Furthermore, we also propose two meta-score metrics, obtained through the combination of single ones. These metrics, which are devised using evolutionary algorithms, are able to improve the single ones. The study of different alignment scoring metrics for the selection of the best alignment proved that in general the best ones are the two proposed meta-scores, followed by the ones that use structural information. Finally, the results obtained demonstrated that MTA with Clustal-W and T-Coffee considerably improves the quality of the alignments.

Acknowledgements

I would like to acknowledge everybody who supported me during the course of this thesis. First and foremost, I want express my gratitude to my supervisors, Dr. Fernando Cores Prado and Dr. Fernando Guirado Fernández, for their continued encouragement and invaluable suggestions throughout this work. They have patiently supervised every little issue, always guiding me in the right direction. Without their help, I could not have finished my thesis successfully.

Special thanks are due to all the seniors from the Group of Distributed Computing (GCD) at the Universitat de Lleida (UdL). I must say that has been a great experience working closely with such good people: Concepció Roig, Francesc Giné, Francesc Solsona, Josep Ll. Lèrida, Josep M. Solà, Valentí Pardo and Xavier Faus.

During my research, I have shared great moments with all my colleagues: Josep Rius, Ignasi Barri, Ivan Teixidó, Héctor Blanco, Anabel Usié, Damià Castellà, Alberto Montañola, Jordi Vilaplana, Eloi Gabaldon, Jordi Lladós and Ismael Arroyo. Many thanks to all of them for the wonderful times we shared.

Furthermore, I want to thank Porfidio Hernández and Toni Espinosa from Universitat Autònoma de Barcelona (UAB) for all their support. I also want to thank all the members of the Comparative Bioiformatics group from the Centre for Genomic Regulation (CRG), especially Cedric Notredame, Carsten Kemena and Paolo Di Tommaso, for their good advice and help, and for welcoming me as if I were one more during one month. Finally, I want to thank all the members of the Edinburgh Data-Intensive Research group at the University of Edinburgh. During three month, I had the chance to stay in this group and work with great people who made me feel at home. It was a great experience that marked me in many positive aspects. I would especially like to acknowledge Malkolm Atkinson, Paolo Besana, David Rodríguez, Rosa Filgueira and Gary McGilvary for all their support and for welcoming me as if I were another member of a big family.

Last but not least, I would like to dedicate this thesis to all my friends and, foremost, to my closest family. Especially to my father Miquel and my sister Marta for the patience and understanding they have shown during these long years. To my girlfriend Laura for her emotional support. And especially, I want to dedicate this thesis to my mother Elena who unfortunately is no longer with us.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Over the twentieth century, biology played a key role among the sciences. A major reason for the growth of biological sciences, was its links to medicine. Progress in medicine depends on fundamental progress in solving the biological challenges. The sequencing of insulin inaugurated the modern era of molecular and structural biology. In this pre-computer era, sequences were analysed and compared by writing them out on pieces of paper. As soon as computers became available, the first computational biologists started to enter these manual algorithms into the memory banks. The application of computer science to answering the questions posed by biologists is called Bioinformatics or Computational Biology. For this reason, bioinformatics can be defined as the computational branch of molecular biology [CN06].

Computational biology has been revolutionized by advances in both computer hardware and software algorithms. Examples include assembling the human genome and using gene-expression chips to determine which genes are active in a cell. High-throughput techniques for DNA sequencing and analysis of gene expression have led to exponential growth in the amount of publicly-available genomic data. For instance, the genetic sequence information in the National Center for Biotechnology Information's GenBank database [BKML+11] has nearly doubled in size each year for the past decade, with more than 37 million sequence records as of August 2004. Biologists are keen to analyse and understand this data, since genetic sequences determine biological structure, and thus, the function of proteins. Understanding the function of biologically active molecules leads to an understanding of biochemical pathways and disease-prevention strategies and cures, along with the mechanisms of life itself.

Computing systems are now so powerful that it is possible for researchers to consider modelling the folding of a protein or even the simulation of an entire human body. As a result, computer scientists and biomedical researchers face the challenge of transforming data into models and simulations that will, for the first time, enable them to gain a profound understanding of the deepest biological functions. Traditional uses of High-Performance Computing (HPC) systems in physics, engineering, and weather forecasting involve problems that often have well-defined and regular structures. In contrast, many problems in biology are irregular in structure, are significantly more challenging for software engineers to parallelize, and often involve integer-based abstract data structures. Solving biological problems may require HPC due either to the massive parallel computation required to solve a particular problem or to algorithmic complexity that may range from difficult to intractable. Many problems involve seemingly well-behaved polynomial time algorithms (such as all-to-all comparisons), but have massive computational requirements due to the large data sets that must be analysed. For example, the assembly of the human genome in 2001 from the many short segments of sequence data produced by sequence robots required approximately 10,000 CPU hours.

Other problems are compute-intensive due to their inherent algorithmic complexity (such as protein folding and reconstructing evolutionary histories from molecular data). Some are known to be NP-hard (or harder). Thus, while NP-hard problems are thought to be intractable, HPC may provide sufficient capacity for evaluating bio-molecular hypotheses or solving more limited, but meaningful, instances.

Among all the topics of molecular biology, the present thesis is focused on Multiple Sequence Alignment (MSA). MSA is one of the techniques of Sequence Alignment, which is the most common task in bioinformatics. Sequence alignment consists of comparing DNA, RNA or protein sequences to search for evolutionary or functional relationships or differences between them [Mou04]. The growth of data, increasing the number of new sequences to be compared and the necessity of aligning DNA sequences, which are longer than protein sequences; meant that many of the algorithms became obsolete because of the complexity problem of aligning multiple sequences, the huge memory consumption or the long runtime.

Many of the tools currently used were created by biologists dealing with data sets that were minuscule in comparison with those available today. As a result, software that was once perfectly adequate now performs slowly or is incapable of successful analysis. As life scientists and biomedical researchers learn more about the complexities of protein structures, computational scientists find that the accurate sequencing of a large number of sequences may be intractable without PetaFLOPS-class computers. When algorithm engineering tools and practices are complemented by high-performance software implementations designed for parallel platforms, enormous gains will be realized in the size of the data sets that may be analysed and the speed in which that analysis is carried out. Therefore, nowadays, the use of HPC systems and techniques is essential for solving MSA.

In this chapter, the concepts of bioinformatics, sequence alignment and HPC are introduced. In Section 1.1, the most relevant concepts of molecular biology and the major research areas are explained to give the reader a basic knowledge to be used during throughout manuscript. Section 1.2 introduces the user to Sequence Alignment research, its applications, requirements and constraints. Then, in Section 1.3, there is a brief summary of HPC to introduce the reader to parallel and distributed systems as some of the most widely-used techniques. Next, once the main terminology and concepts that make up the context of this thesis have been explained, we present its main challenges and motivations. Finally, this chapter ends with the main objectives of the dissertation and details of how this manuscript is structured.

1.1 Bioinformatics

In general terms, bioinformatics is an interdisciplinary area that uses computers to solve problems in molecular biology. Therefore, the field of bioinformatics involves the analysis and interpretation of various types of data, such as nucleotide and amino acid sequences, protein domains, and protein structures. The actual process of analyzing and interpreting data is referred to as computational biology. Important sub-disciplines within bioinformatics and computational biology include the development and implementation of tools to enable efficient access to, use and management of various types of information, and the development of new algorithms and statistics with which to assess relationships among members of large data sets.

In summary, bioinformatics entails the creation and advancement of databases, algorithms, computational and statistical techniques and theory to solve formal and practical problems arising from the management and analysis of biological data.

1.1.1 Molecular Biology

As defined above, bioinformatics is the computational branch of molecular biology. For this reason, in this introduction, it is necessary to clarify some molecular biology concepts.

Molecular biology is the branch of biology that deals with the molecular basis of biological activity. This field overlaps with other areas of biology and chemistry, particularly genetics and biochemistry. Molecular biology is used to understand the interactions between the various systems in a cell, to include the interactions between different types of DNA, RNA and protein biosynthesis, as well as learning how these interactions are regulated.

In previous definitions, some molecular agents and processes were found. In this section, all these concepts needed to understand the central dogma of molecular biology, and used in this present thesis, are briefly explained.

1.1.1.1 Biological preliminaries

Before introducing the central dogma of molecular biology and the main topic of this thesis, the following preliminary biological concepts should be explained:

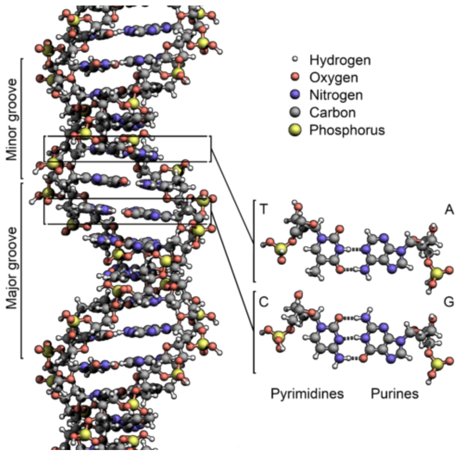
- **Cell:** This is the basic structural and functional unit of all known living organisms. It is the smallest unit that is classified as a living thing, except for the viruses.

- **Nucleic Acid sequences:** Nucleic acid sequences are better known as Deoxyribonucleic acid (DNA) and Ribonucleic acid (RNA).

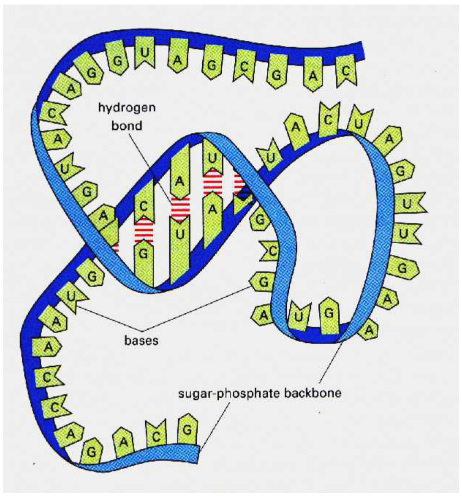
  - **DNA sequences:** These are chains of informational nucleic acids molecules found in the nucleus, which encode the genetic instructions used in the development and functioning of all known living organisms and many viruses. Their main function is the long-term storage of information and they can be seen as the complete blue-print of life, or the whole history book of evolution of life. These molecules are packaged in long units known as **chromosomes**. DNA is made up of four types of molecules, which are codified with four letters: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). As Figure 1.1a shows, most DNA molecules are double-stranded helices. However, DNA is read from one strand alone, as the other side can be predicted.

  - **RNA sequence:** Like DNA, RNA molecules are nucleic acids, but their purpose is to copy information from DNA selectively and to bring it out of the nucleus. RNA serves as a messenger, which delivers the DNA's genetic information to the place where proteins are made. RNA contains four types of molecule: Adenine (A), Cytosine (C), Guanine (G) and Uracil (U). Like DNA, RNA is assembled as a chain of nucleotides, but is usually single-stranded (Figure 1.1b).

- **Protein sequences:** Proteins are large biological molecules manufactured by cells, which consist of one or more chains of amino acids. Proteins perform many functions within living organisms, including catalyzing metabolic reactions, replicating DNA, responding to stimuli, and transporting molecules from one location to another. Protein sequences are shorter than most DNA sequences (an average of 100 characters against 10000 characters) and are built from series of up to twenty different amino acids called residues, represented by letters: Alanine (A), Arginine (R), Asparagine (N), Arspartic Acid (D), Cysteine (C), Glutamine (Q), Glutamic acid (E), Glycine (G), Histidine (H), Isoleucine (I), Leucine (L), Lysine (K), Methionine (M), Phenylalanine (F), Pro-



(a) DNA double helix structure



(b) RNA structure

Figure 1.1: Examples of DNA and RNA sequences

line (P), Serine (S), Threonine (T), Tryptophan (W), Tyrosine (Y) and Valine (V). Proteins sequences, like the one shown in Figure 1.2, are not linear chains of amino acids, and are twisted and folded into three-dimensional structures forming a unique shape. These shapes are studied at four levels: the primary, secondary, tertiary and quaternary.

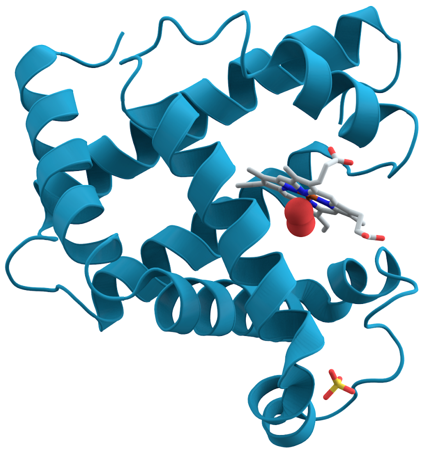


Figure 1.2: 3D structure of the myoglobin protein

- **Chromosomes:** This is an organized structure of DNA and proteins found in cells. For example, humans have 23 pairs of chromosomes.

- **Genes:** A gene is a molecular unit of heredity of a living organism. Genes are lengths of DNA that are specific regions of the genome spread throughout the genome, sometimes contiguous, many times non-contiguous. They are essential for specific functions, either in the development of the organism or in maintaining a normal physiological function. A gene carries biological information in a form that must be copied and transmitted from each cell to all its progeny. The set of genes of a species is its genome. Figure 1.3 shows an example of a gene that occupies a specific location in a chromosome.

- **Genome:** This is all the genetic material contained in the chromosomes of a particular organism and represents all its hereditary information. The genome includes both the genes and the non-coding sequences of the DNA/RNA. For example, the human genome is composed of about 3,000 millions on nucleotides (A, C, T and G) and about 20,000 genes.

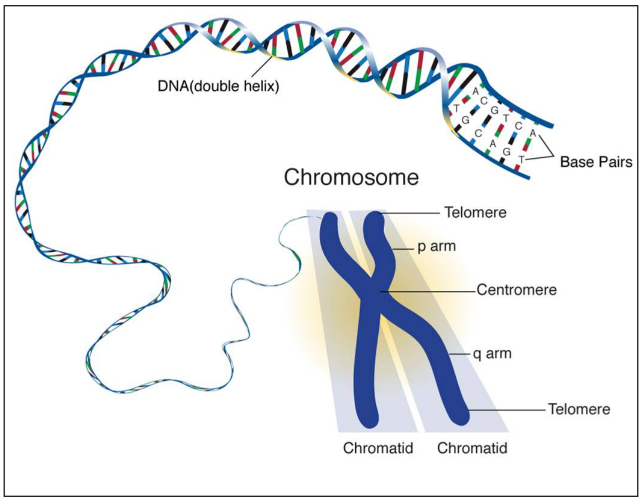


Figure 1.3: A gene that occupies a specific location in a chromosome

1.1.1.2 Central Dogma of molecular Biology

The central dogma of molecular biology was first stated by Francis Crick in 1958 [Cri58]. The dogma is a framework for understanding the transfer of sequence information between DNA, RNA and protein. The dogma classifies the transfer into three types: general transfers, special transfers and unknown transfers.

The general transfers, which are the most common and whose scheme is shown in Figure 1.4, describe the flow of genetic information within a biological system: DNA is duplicated into another DNA (DNA replication); then, this DNA information is transcribed into the mRNA (transcription); and finally, the mRNA travels to the protein production sites and it is synthesized into proteins (translation) [Cri70]. The main operations in this biological process, such as transcription, translation and duplication, are now introduced:

- **Transcription:** The goal of transcriptions is to make an RNA copy of a gene. Specifically, transcription is the process by which the gene information is transferred to a newly assembled piece of messenger RNA (mRNA). This RNA can direct the formation of a protein or be used directly in the cell.
- **Translation:** This is the process that converts an mRNA sequence into a chain of amino acids that form a protein.
- **DNA Duplication:** DNA replication is the process of copying a double-stranded molecule. DNA duplication is necessary to transmit the genetic information between parents and progeny and repeat the transference cycle.

1.1.2 Bioinformatic Research Areas

The primary goal of bioinformatics is to increase the understanding of biological processes. To achieve this goal, bioinformatics is focused on developing and applying computationally intensive techniques, such as pattern recognition, data mining, machine learning algorithm, HPC algorithms and visualization. In this subsection, the major search areas are presented. These are

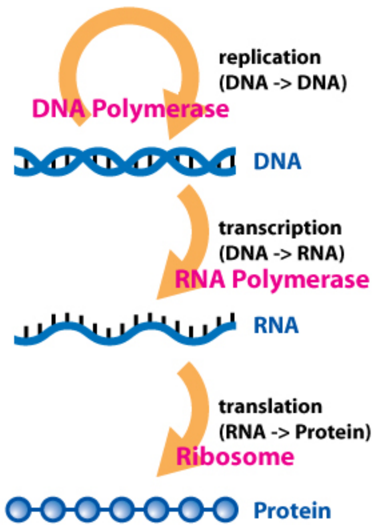


Figure 1.4: Central dogma of molecular biology scheme

where computer science, mathematics and statistics have been used to define algorithms methods and systems capable of finding solutions or facilitating the research.

- **Sequence analysis:** This consists of comparing DNA, RNA or protein sequences to understand their features, function, structure, or evolution [DEKM98]. Comparing new sequences to those with known functions is a key way of understanding the biology of an organism from which the new sequence comes. The most common sequence analysis methodologies are sequence alignment and searches against biological databases. In molecular biology, sequence analysis includes the following relevant topics:
  - Comparison of sequences in order to find similarity between sequences and determine if these are related (homologous sequences). Two or more sequences are homologous when two protein or gene sequences have the same ancestor, similar functions and similar structures.
  - Identification of intrinsic features of the sequence, such as active sites, post-translational modification sites, gene-structures, reading frames, distributions of introns and exons and regulatory elements.

  - Identification of sequence differences and variations, such as mutations and single nucleotide polymorphism (SNP), in order to obtain the genetic marker.
  - Revealing the evolution and genetic diversity of sequences and organisms.
  - Identification of molecular structure from isolated sequences.

  Nowadays, there are many tools and techniques that provide sequence alignments and analyze the alignments produced to understand their biology.

- **Genome annotation:** This is the process of marking the genes and other biological features in DNA sequences. This process is divided into three steps: First, the genome portions that do not code for proteins are identified. Then, the gene prediction or gene finding is done. This consists of identifying elements in the genome. Finally, the biological information is attached to the sequences.
- **Computational evolutionary biology:** This is the study of the origin and descen of species, as well as their changes over time. It consists of building the tree of life that determines evolutionary relationships between species.
- **Literature analysis:** This consists of using the computational and statistical linguistics to explore the huge number of published literature.
- **Gene expression:** This is the process whereby the information from a gene is used in the synthesis of another functional gene product, normally a protein. This process is used by all known life forms to generate macromolecular machinery for life. Micro-arrays are tiny chips used to study gene expression.
- **Gene regulation:** This is a set of mechanisms used by cells to increase or decrease the production of specific gene products (proteins or RNA). Bioinformatic techniques have been applied to exploring various steps in this process.

- **Protein expression:** This is a subcomponent of gene expression that consists of the stages after DNA has been transcribed into messenger RNA (mRNA). The mRNA is then translated into polypeptide chains, which are ultimately folded into proteins. Protein expression is commonly used by researchers to denote the measurement of the presence and abundance of one or more proteins in a particular cell or tissue. Protein micro-arrays and High Throughput (HT) Mass Spectrometry (MS) [NMA+10] are techniques used in protein expression.

- **Comparative genomics:** This is the study of the relationship of genome structure and function across different biological species or strains.

- **Modeling biological systems:** This consists of developing and using efficient algorithms, data structures, visualization and communication tools with the goal of modeling biological systems with computers. It involves the use of computer simulations of biological systems, like cellular subsystems to both analyze and visualize the complex connections of the cellular processes.

- **High-throughput image analysis:** This consists of using the computational technologies to accelerate or fully automate the processing, quantification and analysis of large amounts of high-information-content biomedical imagery.

- **Protein structure prediction:** This is the prediction of the three-dimensional structures of a protein from its amino acid sequence. It consists of predicting the secondary, tertiary, and quaternary structure from its primary structure.

- **Molecular interaction:** It is the study of the interactions among proteins, ligands and peptides. For studying molecular interactions, docking algorithms [LR96] are efficient algorithms, which consist of the dynamic molecular simulation of the movement of atoms around rotatable bonds.

1.2 Sequence Alignment

Nowadays, sequence alignment is by far the most common task in bioinformatics. Given a set of DNA, RNA or protein sequences, sequence alignment is a way of arranging the residues of these sequences to identify similarity or divergence regions that could be consequence of functional, structural or evolutionary relationships between sequences [Mou04].

Sequences are aligned to visualize the effect of evolution across sequences that share a common ancestor. Similar regions in homologous protein sequences, called conserved regions due to the lack of substitutions or the presence of only very conservative substitutions, determine that this region is of structural or functional importance. In DNA and RNA sequences, the conservation can mean a similar functional or structural role. Otherwise, the precise meaning of equivalence is generally context dependent: for the phylogeneticist, equivalent residues have common evolutionary ancestry; for the structural biologist, equivalent residues correspond to analogous positions belonging to homologous folds in a set of proteins; for the molecular biologist, equivalent residues play similar functional roles in their corresponding proteins.

On the other hand, mismatches between sequences can be interpreted as insertion or deletion mutations. Mutations are changes in sequences and are caused by radiation, viruses, transposons and mutagenic chemicals, errors that occur during cell transcription process, and can also be induced by the organism itself. Insertion mutations are the addition of one or more bases or residues into sequences, whereas deletions or gaps are mutations in which some bases or residues are missing.

Alignments are represented both graphically and in text format. As Figure 1.5 shows, the sequences are written in rows arranged so that aligned residues appear in successive columns. Gaps are spots in sequences where a residue or a segment of residues is missing and is usually represented as indels. Aligned columns containing identical or similar characters are indicated with a system of conservation symbols.

Computational approaches to sequence alignment are classified into three categories: global alignments, local alignments and a hybrid of these.

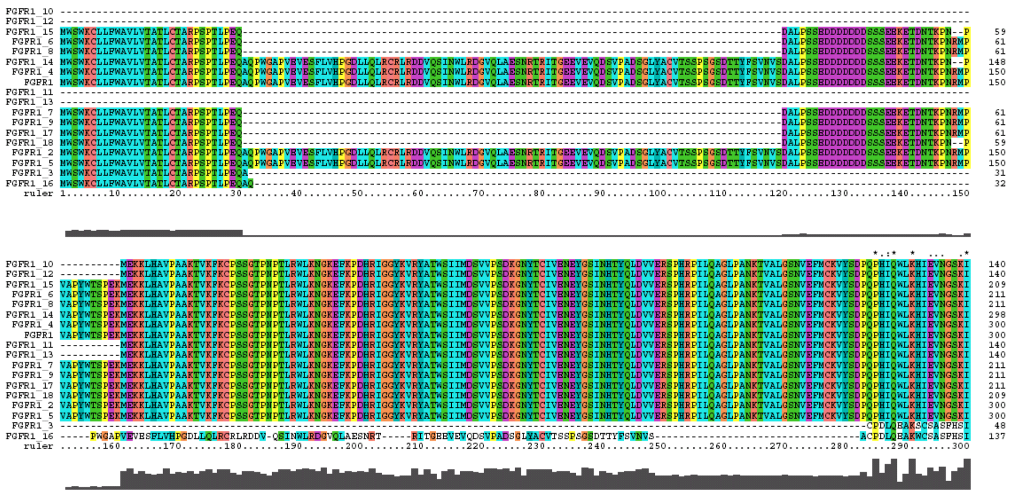


Figure 1.5: MSA representation

- **Global alignments:** Global alignments consist of creating an end-to-end alignment of the sequences by aligning every residue in every sequence (Figure 1.6). They are more useful when the sequences to be aligned are similar in length and similar across their entire lengths. The Needleman-Wunsch algorithm is the general global alignment technique and is based on dynamic programming [NW70].

- **Local alignments:** Local alignments consist of describing the most similar regions within the sequences to be aligned (Figure 1.6). They identify regions of similarity within long sequences that are often widely divergent overall and ignore the other regions. They are more useful for analogous sequences that are suspected of containing regions of similarity or similar sequence motifs. In this case, the Smith-Waterman algorithm is the general local alignment and is also based on dynamic programming [SW81].

- **Semiglobal or glocal alignments:** Glocal alignments are hybrids of the global and local methods. They try to find the best possible alignment that includes the start and end of one or the other sequence [BMP^{+}03].

Another categorization depends on the number of sequences to be aligned.

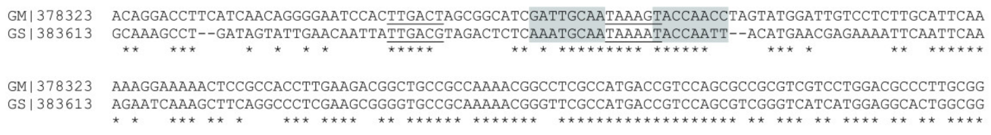
```
Global  FTFTALILLAVAV
        F--TAL-LLA-AV

Local   FTFTALILL-AVAV
        --FTAL-LLAAV--
```

Figure 1.6: Global and local alignment example

Sequence alignments can be classified into two types: Pairwise Sequence Alignments and Multiples Sequence Alignments. Next, these two categories are briefly described, focusing on MSA which is the main topic of the present study.

1.2.1 Pairwise Sequence Alignment

Pairwise Sequence Alignments are global or local alignments of two input sequences. Figure 1.7 shows an example of a Pairwise Sequence Alignment representation. Pairwise Sequence Alignments are efficient to calculate and are often used for methods that do not require extreme precision, such as database searching.



```
GM|378323   ACAGGACCTTCATCAACAGGGGAATCCACTTGACTAGCGGCATCGATTGCAATAAAGTACCAACCTAGTATGGATTGTCCTCTTGCATTCAA
GS|383613   GCAAAGCCT--GATAGTATTGAACAATTATTGACGTAGACTCTCAAATGCAATAAAATACCAATT--ACATGAACGAGAAAATTCAATTCAA
             **   ***     *  *  * *      *****        ** * ********* ******      *** *        **   ******

GM|378323   AAAGGAAAAACTCCGCCACCTTGAAGACGGCTGCCGCCAAAACGGCCTCGCCATGACCGTCCAGCGCCGCGTCGTCCTGGACGCCCTTGCGG
GS|383613   AGAATCAAAGCTTCAGGCCCTCGAAGCGGGGTGCCGCAAAAACGGGTTCGCCATGACCGTCCAGCGTCGGGTCATCATGGAGGCACTGGCGG
            * *   *** ** *    *** ****  ** ****** *******  ******************* ** *** ** **** ** ** ****
```

Figure 1.7: Pairwise Sequence Alignment example

1.2.2 Multiple Sequence Alignment

Multiple Sequence Alignment (MSA) is an extension of Pairwise Sequence Alignment to align more than two sequences at the same time (Figure 1.5). The main objective of MSA is to assemble the sequences reflecting the biological relationship between them. MSA gives the biologist relevant biological information as structural and functional information. Therefore, MSA is used in identifying evolutionary relationships between sequences, searching conserved sequence motifs that play an important role in the function and structures of a group of proteins, and predicting structures [Not02].

1.2.2.1 Applications

In bioinformatics, MSA is a useful tool used to visualize the effect of evolution, identify conserved motifs or predict the secondary and tertiary structure. The most common MSA uses are described below.

- **Phylogenetic analyses:** The MSA technique is essential for phylogenetic tree construction. Phylogenetic trees are used to classify the evolutionary relationships between homologous genes represented in the genomes of divergent species. In short, it determines the evolutionary relationships between among various organisms from a tree that defines the ancestors and descendants of the species.

- **Identification of conserved motifs and domains:** A sequence motif is a nucleotide or amino-acid sequence pattern that is widespread and has biological significance. Motif finding or profile analysis is a method for searching sequence motifs in global MSAs. MSA techniques are used in motif finding to identify motifs preserved by evolution that are essential in the structure and function of a group of related nucleotides or proteins. MSAs and known conserved motifs are powerful tools for characterizing sequences with unknown functions combining these motifs with experimental data.

- **Structure prediction:** This is an important application of MSA because of the close relationship between the structure and the function of a protein. It consists of predicting its secondary and tertiary structure from its primary structure.

1.2.2.2 Constraints and Requirements

Producing a suitable MSA is not trivial and is a complicated problem in both biological and computational issues.

From the biological aspect, it is difficult to ensure the biological correctness of an alignment. Nowadays, methods define an objective function, which defines the mathematical objective of the search. The main problem is that the perfect objective function that defines the mathematically optimal alignment is not guaranteed to be biologically optimal [KN09]. Furthermore, defining this proper objective function is a highly non-trivial task and an important research area.

In the computational issue, the computation of a mathematically optimal alignment is an NP-Complete problem and therefore it is only possible to produce small alignments [WJ94]. From a computational point of view, several ways have been proposed to address the computational problem:

- The first was by developing faster heuristic algorithms that reduce computational space for the most time-consuming tasks. Nowadays, the most popular MSA methods are heuristic or probabilistic. These methods align the sequences by maximizing their similarity and none of them guarantees a full optimization. However, the new data-intensive era has made many of these heuristics obsolete due to runtime and memory constraints.

- The second is running these algorithms on specialized chips (bio-accelerator). The runtime of progressive alignment programs is clearly dominated by the first step (computation of pairwise sequence distance). There are two basic approaches to parallelizing this step: one is based on the systolisation of the pairwise distance computation algorithm (fine-grained); the other is based on the distribution of the computation of pairwise distances (coarse-grained). Systolic array architectures have proved their high efficiency for pairwise sequence alignment using dynamic programming [AEAT+07].

- The third and the HPC solution consists of applying parallel computing to these algorithms. The problem can be divided into smaller tasks and processed in parallel by two or more microprocessors that can be used simultaneously. This approach treats the MSA method from a coarse-grained focus.

1.2.2.3 Evaluation

The development of new MSA algorithms raises the need for an efficient way to evaluate the alignment accuracy, in order to select the best alignment among those produced by the available algorithms.

Nowadays, the common methods for validating MSA methods consist of comparing an alignment against a reference alignment using reference sequence alignment databases, such as Balibase [TPP99], Homstrad [MDBO98], Prefab [Rob04] and SABmark [WLW05]. The benchmarking databases are usually constructed using 3D protein structural alignments, and they are thus independent of sequence alignment methods [EB06]. The quality of the MSA programs has typically been assessed by a mathematical function that respectively measures the proportion of correctly aligned residue pairs or alignment positions in the alignment [TPP99]. However, the need for a reference alignment is a limitation for validating unknown sequences. For this reason, these databases are used to validate or compare MSA applications.

Another way of evaluating alignments is to use an objective function as a similarity measure. This method consists of maximizing the objective function, which is based on sequence identity. These scoring metrics should incorporate everything that is known about the sequences, including their structure, function and evolutionary history to make them more accurate. However, as is known, a perfect scoring metric can determine the mathematically optimal alignment, but this is not always the biologically optimal one.

1.2.2.4 Current Status

Due to the entry into the area of comparative genomics, the simultaneous comparison of a large number of homologous sequences has become more and more important and there is no doubt that MSA has come to play a key role in molecular biology.

The number of available MSA methods has increased over the last 30 years. However, only a minority of the methods proposed in the literature is regularly used. The main reason for their failure is that there is no satisfactory theoretical framework in sequence analysis and improvements are driven by results and not theory.

Over the last ten years, MSA has undergone drastic evolutionary changes with the introduction of several algorithms and evaluation methods. Instead of the traditional progressive alignment, current trends are based on the use of iterative optimization strategies, consistency-based scoring schemes and the integration of heterogeneous information, such as structures, results of database searches, experimental data, etc. Due to the integration of this information, new MSA methods are memory and CPU hungry.

For these reasons and the explosion of new data due to the genomic revolution, currently known and future methods should be optimized and must evolve to take new faster computational techniques and distributed systems into account.

1.3 High Performance Computing

The term High-Performance Computing (HPC) can be defined as the use of parallel processing for running advanced application programs efficiently, reliably and quickly.

The history of HPC or Supercomputing dates back to the 1960s when a series of computers at Control Data Corporation were designed by Seymour Cray to use innovative designs and parallelism to achieve superior computational peak performance [Che09]. A supercomputer is a computer on the cutting edge of current processing capacity, particularly speed of calculation. While the supercomputers of the 1970s used only a few processors; in the 1990s, machines with thousands of processors began to appear, and by the end of the 20th century, massively parallel supercomputers with tens of thousands of "off-the-shelf" processors were the norm [HT89][HJS00]. Supercomputers in the 21st century can use over 100,000 processors, such as CPU or GPU, connected by fast networks.

Although, the HPC community has its roots in solving computational problems in physics (such as fluid flow, structural analyses, and molecular dynamics); traditional approaches to these problems, and to ranking HPC systems based on the Linpack benchmark, may not be the optimal approach to HPC architectures in computational biology. Many researchers are carefully considering the architectural needs of HPC systems to enable next-generation biology. New HPC algorithms for biomedical research will require close integration of computation with database operations and queries, along with the ability to handle new types of queries that are highly dependent on irregular spatial or temporal locality.

In this section, parallel and distributed computing systems, categorizations and applications are briefly described.

1.3.1 Parallel Computing

Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are solved concurrently [Fos95]. It can also be defined as the simultaneous use of multiple processing elements to solve a problem. Although parallelism has been used in HPC for many years, the physical constraints preventing frequency scaling have led parallel computing to become the dominant paradigm in computer architecture. Currently, parallel computing is used in a wide range of fields, from bioinformatics (protein folding and sequence analysis) to economics (mathematical).

A parallel application represents a huge problem that has been divided into multiple parts or tasks, so that each processing element can execute its task in the algorithm simultaneously with the others. These tasks, called threads or processes, can be completely independent or have dependencies and hence, communication with other tasks. Parallel applications are often classified according to how often their subtasks need to communicate with each other, or the relative size of the units of computation that execute in parallel (granularity).

- **Fine-grained parallelism:** Application tasks share many dependencies; therefore, they must communicate many times per second and the units of computation are small.

- **Coarse-grained parallelism:** Application tasks share few dependencies and do not communicate many times per seconds and thus the granularity is bigger.

- **Embarrassingly parallel:** Application tasks without dependencies that rarely or never have to communicate. They are considered the easiest to parallelize.

Common knowledge gained from working on parallel applications suggests that obtaining an efficient parallel implementation is fundamental to achieving a good distribution of both data and computations. In general, any parallel strategy represents a trade off between reducing communication time and improving the computational load balance. This balance is achieved by choosing the most suitable task scheduling strategy for the problem.

Moreover, as Amdahl's law states [Amd67], the improvement in performance gained by the use of parallel computing depends very much on the software algorithms used and their implementation. In particular, possible gains are limited by the fraction of the software that can be run in parallel simultaneously on multiple processors.

Another categorization is Flynn's taxonomy classification [Fly72]. Flynn classified programs and computers by whether they were operating using a single or multiple sets of instructions, whether or not those instructions were using a single or multiple sets of data.

- **Single Instruction Singe Data (SISD):** This classification is equivalent to an entirely sequential program.

- **Single Instruction Singe Multiple (SIMD):** This is analogous to doing the same operation repeatedly over a large data set. SIMD machines consist of many simple processors, each with small local memory. Every processor must execute the same instruction with different data in each computing cycle. When a processor needs data stored on another processor, an explicit communication must pass between them to transfer this data to the local memory. The complexity and often the inflexibility of SIMD machines, strongly dependent on the synchronization requirements, have restricted their use mostly to special-purpose applications, such as signal processing applications.

- **Multiple Instruction Single Data (MISD):** This is a rarely used classification. While computer architectures have been devised to deal with this, such as systolic arrays, few applications that fit this class are materialized.

- **Multiple Instruction Multiple Data (MIMD):** This is by far the most common type of parallel program. In MIMD machines, each computational process executes at its own rate in an asynchronous fashion with complete independence of the other computational processes [HX98]. The memory architecture has a strong influence on the global architecture of MIMD machines, becoming a key issue for parallel execution, and frequently determines the optimal programming model.

1.3.1.1 Hardware

After introducing the term parallel computing and its main characteristics and classifications, it is necessary to briefly describe and categorize the architectures most commonly used for running parallel applications.

First of all, an important classification of parallel architectures is related to the main memory. The main memory in a parallel computer is either shared memory or distributed memory.

- **Shared memory:** A system is said to have shared-memory architecture if any process, running in any processor, has direct access to any local or remote memory in the whole system. Shared memory architecture brings several advantages to bioinformatics applications as a single address map simplifies the design of parallel programs. In addition, each element of the main memory can be accessed with equal latency and bandwidth. For these reasons, this kind of architecture is known as Uniform Memory Access (UMA). The main disadvantage of shared memory is that it does not scale well as the number of processors in the computer increases.

- **Distributed memory:** It refers to the fact that the memory is logically distributed and each processor has its own private memory. These architectures, which are known as Non-Uniform Memory Access (NUMA), scale very well. On the other hand, the lack of a single physical address map for the memory incurs a time penalty for inter-process communication.

This memory and communications classification introduces the concept of distributed computing. Sometimes, the terms of parallel computing and distributed computing overlap and there is no clear distinction between them. However, parallel computing may be seen as a particular tightly-coupled form of distributed computing, and distributed computing may be seen as a loosely-coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems roughly as parallel or distributed using the following criteria:

- In parallel systems, all processors have access to a shared memory. Shared memory can be used to exchange information between processors. An example of a parallel systems is multicore processors or symmetric multiprocessors (SMP).

  A multicore processor is a processor that includes multiple execution units, known as cores, on the same chip. A multicore processor can issue multiple instructions per cycle from multiple instruction streams. On the other hand, a symmetric processor is a computer system with multiple identical processors that are connected to a single shared main memory and are controlled by a single bus. Most common multiprocessor systems today use an SMP architecture. In the case of multicore processors, the SMP architecture applies to the cores, treating them as separate processors.

- In distributed systems, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors. Such systems include computing clusters, Grids and global computing systems gathering computing resources from individual PCs over the Internet. There are many reasons to explain the continuous growth of distributed systems and distributed computing. Their main characteristics and systems are explained below.

Distributed systems

Distributed computing is formally defined as “a computer system in which several interconnected computers share the computing tasks assigned to the system [IEE91]".

A distributed system consists of multiple computers that communicate and coordinate their actions through a computer network by passing messages [DKC05]. Those computing technologies have recently emerged as new paradigms for solving complex computational problems. These systems enable large-scale aggregation and sharing of computational data and other geographically distributed computational resources. In recent years, many researchers have reported numerous advances and innovative techniques for such paradigms, from theoretical design to the application development. Another fact that has contributed strongly to the rapid development of large-scale applications in many fields of science and engineering is the continuous development of high-speed networks and more especially, Internet.

The use of a distributed system in other applications would be beneficial for many reasons. For example, in comparison with a single high-end computer, it may be more cost-efficient to obtain better performance by using a cluster of several low-end computers to avoid memory contention. As a distributed system has no single point of failure, it can also be more reliable than a non-distributed system. A distributed system may even be cheaper, and easier to expand and manage, than a supercomputing system with the same performance.

Inside the category of distributed systems, one can find a wide range of systems. Next, the most popular and those most closely related to the field of interest of this thesis are briefly explained.

- **Cluster computing:** A cluster is defined as a collection of computing resources that consists of a set of loosely independent connected computers that work together so that in many aspects they can be viewed as a single system. As Figure 1.8 shows, the cluster nodes are usually interconnected to each other through fast local area networks (LAN) on a single point, often using a switch. Each computing node runs its own

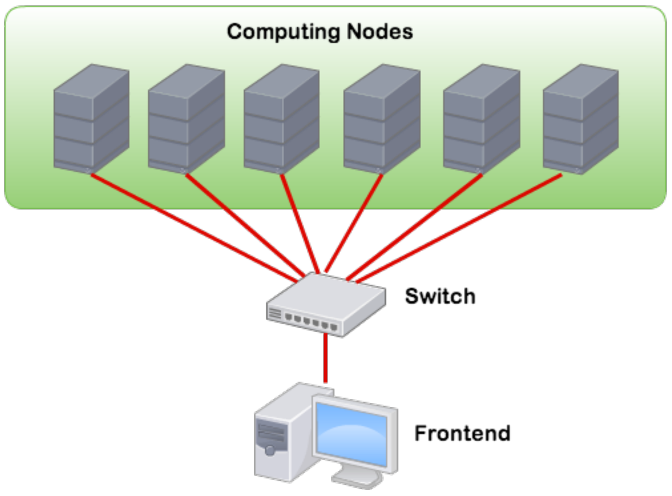


Figure 1.8: Cluster environment structure

instance of the operating system and these are supervised within a single administrative domain, usually residing in a single room and managed as a single computer system.

Computer clusters emerged as a result of the convergence of a number of computing trends, including the availability of low-cost microprocessors, high-speed networks, and software for high-performance distributed computing. Furthermore, not only are clusters made up of multicore desktop PCs or workstations, but they can also take advantage of the resources and properties obtained through such specialized hardware as Graphics Processors Units (GPU), Field-Programmable Gate Arrays (FPGA) or Vector processors.

Clusters are developed to improve performance and availability compared with single computer. Moreover, clusters can be classified according to their processing power, with these ranging from small business clusters with few nodes to clusters whose performance reaches or surpasses the fastest supercomputers in the world.

- **Multi-Cluster computing:** A multi-cluster can be defined as the union of a set of clusters using a dedicated communications network within an organization or institution.

The main architectural feature that differentiates a multi-cluster from other distributed computing environments, such as Grid or Peer-to-Peer, is that the communications networks between clusters are connected by dedicated links. This has two important implications: The first is that multi-clusters have a reliable and predictable bandwidth between the resources of different clusters [JAA06], unlike distributed systems, which are connected over the Internet. The second is that the availability of the resources are more predictable and controllable in multi-cluster systems, while the rest of these resources systems are very dynamic and changeable, so nobody can guarantee their availability at any specific time.

Figure 1.9 shows the basic architecture of a multi-cluster environment, where a set of clusters is connected to a central switch by dedicated network links.

- **Grid computing:** Grid is a distributed paradigm that appeared many years ago as a real alternative to the expensive supercomputers for HPC. Grid computing is defined as "coordinated resource sharing and problem solving in large, multi-institutional virtual organizations". As in Figure 1.10 shows, a Grid system is a virtual joining of a large number of separate computers, clusters or LANs workstations connected by a network, usually the Internet.

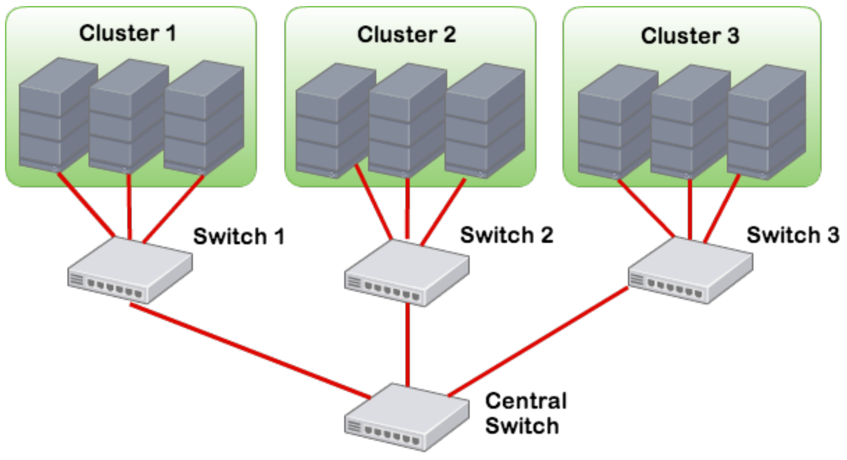


Figure 1.9: Multi-Cluster environment structure

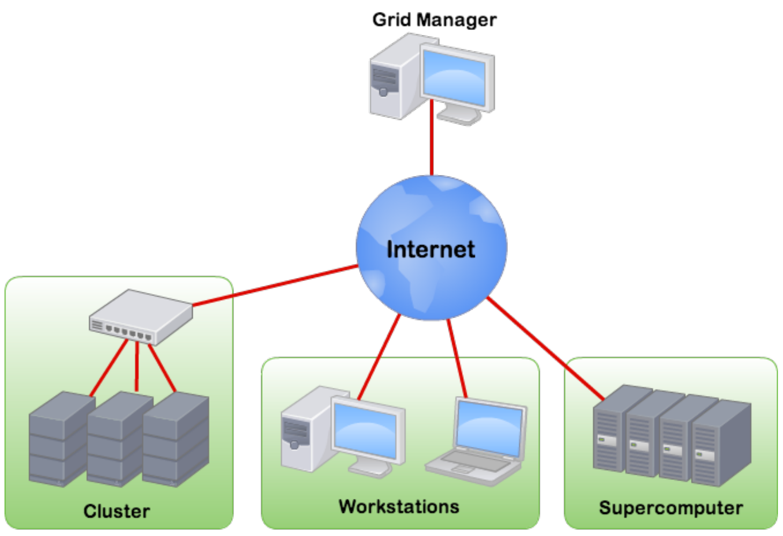


Figure 1.10: Grid environment structure

One advantage of this distributed system is that it allows a larger amount of computation that would be possible on a single computer or a single cluster within a single institution. However, grid computing has some disadvantages, such as the high complexity of management and administration software (such as the Globus Toolkit [Fos05]) and the high management and maintenance cost.

- **Peer-two-Peer computing (P2P):** P2P computing originated as a new paradigm after the traditional client-server computing that became very popular for file sharing among internet between users (Napster, Gnutella, BitTorrent, etc).

  A P2P system is defined as a computer network in which each computer, called node or peer, in the network can act as a client or server for the other computers in the network, allowing shared access to various resources, such as files, computing resources and sensors without the need for a central server.

  As Figure 1.11 shows, peers share a portion of their computational resources with other peers through Internet, without a need for central coordination by servers or stable hosts. In this kind of distributed paradigm, it may be possible to construct cheap distributed systems on

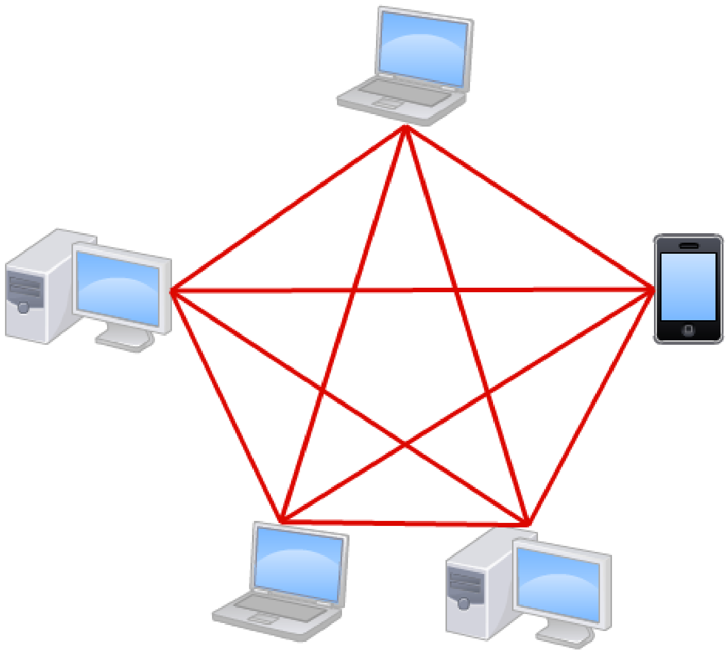


Figure 1.11: P2P Network

Internet for parallel/distributed processing with management and energy consumption at almost zero cost.

- **Cloud computing:** Cloud computing is a new paradigm that has rapidly spread in recent time and consists of the use of computing resources (hardware and software) that are delivered as a service over a network, typically Internet. It is a multi-purpose paradigm that enables efficient management of data centers, time-sharing and virtualization of resources with a special emphasis on business models. The name comes from the use of a cloud-shaped symbol, like the one shown in Figure 1.12, as an abstraction for the complex infrastructure it contains in system diagrams.

  Clouds can be viewed as a logical continuation from Grids by providing a higher-level of abstraction [JMF09]. Therefore, the major idea of this paradigm is the existence of infinite computing resources available on demand and the ability to pay for the use of computer resources on a short-term basis as needed. This allows companies to save costs by

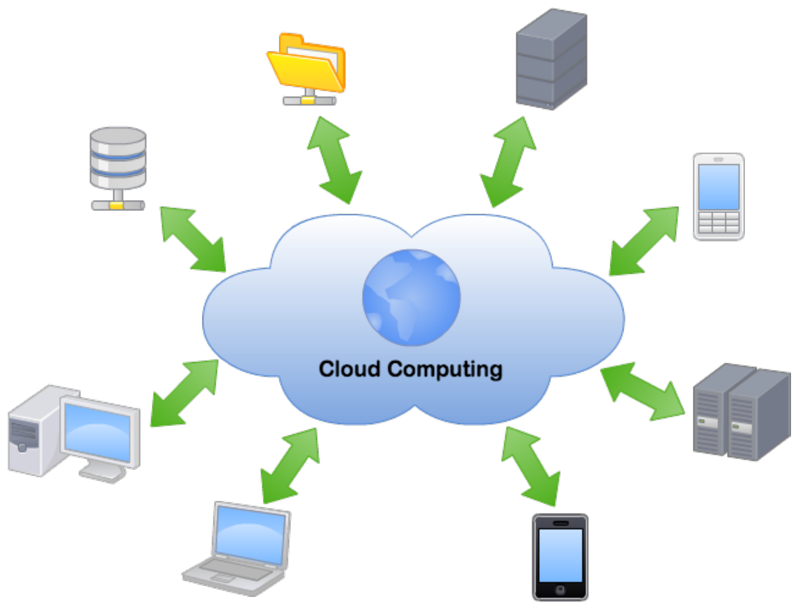


Figure 1.12: Cloud environment structure

having a limited set of resources that can be increased according to their needs [AFG+09].

1.3.1.2 Software

The growth of parallel programming has led to the creation of many concurrent programming languages, libraries, APIs, parallel programming models and new task scheduling strategies.

Parallel languages can generally be divided into classes based on the memory architecture: shared memory or distributed memory.

- Shared memory programming languages communicate by manipulating shared memory variables. POSIX Threads and OpenMP are two of most widely used shared memory APIs.

- Distributed memory uses message passing between the computing nodes. In this case, MPI and PVM are the most common ones for message-passing system APIs.

Furthermore, there are specialized APIs for programming specialized systems, such as CUDA or OpenCL for the Graphics Processor Unit (GPU). The graphics processing unit (GPU) has become an integral part of today's mainstream computing systems. Over the past six years, there has been a marked increase in the performance and capabilities of GPUs. The modern GPU is not only a powerful graphics engine but also a highly-parallel programmable processor featuring peak arithmetic and memory bandwidth that substantially outpaces its CPU counterpart. The GPUs' rapid increase in both programmability and capability has spawned a research community that has successfully mapped a broad range of computationally demanding, complex problems to the GPU. This effort in general-purpose computing on the GPU, has positioned the GPU as a compelling alternative to traditional microprocessors in high-performance computer systems of the future [OHL+08]. Thus, in the fields of sequencing and protein docking tasks, a large performance benefit is obtained through the use of GPU computing. The amount of available data will grow even further in the near future due to advances in high-throughput technologies, leading to a data explosion. Since GPU performance grows faster than CPU performance, the use of GPUs for Bioinformatics will be a valid option.

1.4 Motivation

Although molecular biology and computer science were born at almost the same time, they have grown explosively as separated disciplines. With the growth of the information culture, efficient digital searches are needed to extract and abstract useful information from massive data. In the biological and biomedical fields, massive data take the form of sequences plain files, 3D structures, motifs, 3D microscopic image files, huge databases, etc. However, while genome projects and DNA arrays technology are constantly and exponentially increasing the amount of data available, the ability to treat and process all this information remains near constant.

Due to the recent evolution in computer processing speed, which has increased exponentially (like some areas of knowledge in molecular biology), HPC systems can handle the growing demand posed by bioinformatics applications. The current literature has demonstrated that parallel computing is an effective way of dealing with some of the hardest problems in bioinformatics. The use of parallel computing schemes expands resources to the size of the problem to be tackled.

Currently, the use of HPC in molecular biology is widespread, and many of the most complex algorithms and heuristics have already been studied, optimized and parallelized for execution in parallel or distributed systems. For instance, NP-Complete dynamic programming algorithms for sequence alignment, such as Needleman and Wunsch algorithm, Smith and Waterman algorithm, and other faster heuristics, have been parallelized. Besides, many database search heuristics, such as those implemented in BLAST or FASTA, have also been optimized to run in parallel. Even with current algorithms and optimized heuristics, sequential programs may run for days, or even weeks, on a larger data set. Therefore, computer scientists and bioinformatics technicians have also focused their work on optimizing and accelerating whole applications, such as MSA methods or phylogenetic tree construction programs.

However, several other challenges in bioinformatics remain unsolved as far as parallel computing is concerned. These problems represent attractive challenges for biologists and computer scientists in the coming years.

With regard to MSA, which is the main topic of the present work, the computation of optimal alignments is an NP-Complete problem, meaning that only a few sequences can be aligned. Therefore, the use of sub-optimal heuristics is more extended. In addition, the exponential growth of data and the size of this, has made it intractable for traditional sequential algorithms that have become obsolete as they simply cannot solve the problem, due to memory constraints, or are unable to do in a reasonable period of time. Moreover, it has been seen that the improvements obtained by traditional parallelization techniques are not enough to enable them to run these algorithms on distributed systems, and sometimes, the problem is one of the design and complexity of these algorithms. Thus, it is necessary for biologists and computer scientists not only to apply parallel computing to these methods, but also to go into detail and study deeply the methods in order to optimize the algorithms to be implemented in parallel and successfully executed in distributed systems.

1.5 Thesis Objectives

As mentioned above, owing to the exponential growth of data and the inability to treat efficiently, the interrelation between biologist, bioinformatics and computer scientist is ever more necessary. Therefore, from the beginning of the present thesis, one of the main objectives was to provide a link between such very different worlds as molecular biology and HPC. The idea is to discuss their problems and share knowledge in order to advance in the design and implementation of algorithms and methods that overcome the existing limitations and take advantage of new computing technologies and increasing computer processing speed.

To achieve this, the present thesis is a collaborative work with a group from the Centre for Genomic Regulation (CRG) from Barcelona. One of the topics this group is working on is sequence alignment. Moreover, they are responsible for the development of T-Coffee [NHH00], a leading MSA application.

T-Coffee, which is explained in greater detail in future chapters, is an accurate multiple sequence aligner based on the progressive alignment heuristic that introduces consistency information into the alignments. The improvements in accuracy penalize its performance, restricting its ability to align many sequences simultaneously and face the challenges of the new data-intensive era. Therefore, the main objective of this thesis was focused on designing and developing alternative techniques and algorithms to solve the studied problems and improve its performance, while trying to avoid affecting the biological accuracy of the method negatively. However, the ultimate goal is not to limit these proposed techniques to a single method, but rather to make it possible to implement them in other programs to achieve similar performance results.

More specifically, to achieve this objective, three different points have been worked on this thesis: Increasing scalability, improving parallelism, and improving accuracy.

1. **Increasing scalability.** It is known that the introduction of consistency information to improve the alignment accuracy increases the CPU and memory requirements exponentially to the number of sequences and its length. Consistency-based schemes reduce the scalability so limiting the

number of sequences the method is able to align. Therefore, one target of this work was to design a technique to reduce the runtime and memory constraints and thus, increase the scalability in order to enable the method to treat more sequences.

2. **Improving parallelism.** Parallel or concurrent implementations have shown scalability problems when the number of sequences increases, due to data dependencies in the alignment process. For this reason, another goal of this thesis was to design a technique to exploit the degree of parallelism in order to reduce the execution time and increase the scalability of the parallel applications.

3. **Improving accuracy.** The accuracy of progressive alignment methods is very dependent on the order in which the sequences are aligned. The last objective was to design and implement a new MSA method, based on the studied methods, to improve the biological accuracy of the alignments.

It is also necessary to say that, before considering possible solutions, the first tasks and objectives of this thesis were to study and analyze the execution time and memory requirements of T-Coffee, its parallel implementation, which is called Parallel-TCoffee, and other MSA methods in order to model the major problems and constraints.

1.6 Overview

This section describes the chapters in this thesis.

- **Chapter 1: Introduction.** This chapter introduces the thesis topic, identifying the research questions and showing how these will be addressed. First, some important bioinformatic concepts are explained to provide readers with some background. Secondly, sequence analysis is introduced and its uses and main drawbacks are explained. Third, the HPC concept is described and some classifications of the main architectures and software used are detailed. After doing so, the motivation and the main objectives of this thesis are presented and the chapter finishes with a quick overview of the following chapters.

- **Chapter 2: Related work.** This chapter reviews the work done in various areas related to the material covered in this thesis. The chapter starts by describing the main features of MSA heuristics and methods. Then, it explains the major HPC work done in the MSA field and its main applications. Finally, it introduces the principal MSA benchmarking tools and evaluating scores.

- **Chapter 3: Balanced Guide Tree.** This chapter presents a new clustering method to increase the degree of parallelism in parallel or concurrent implementations. First, it analyses the application studied and models the problems. Then, it describes the main contributions before presenting and evaluating the experimental results.

- **Chapter 4: Consistency Library reduction.** This chapter presents a new technique for reducing the memory constraints of consistency-based methods in order to improve scalability. Its structure is similar to Chapter 3: First, the problem is introduced, then the proposal is described and finally the experimental results are shown and discussed.

- **Chapter 5: Multiples Trees Alignment.** This chapter presents a new MSA method for improving the accuracy of the alignments. Moreover, it also describes two new meta-scores obtained through genetic algorithms for evaluating the accuracy of the alignments. As in previous chapters, it introduces the final proposals and its main characteristics. Finally, the experimental results are analyzed to discuss and compare them with other proposals in the literature.

- **Chapter 6: Conclusions and Future Work.** This chapter reviews the key points. The objectives and results of the entire research thesis are summarized, special emphasis being placed on the scientific contributions that have been introduced. At the end, some prospective points for the future work on this research are provided. Furthermore, the main publications produced during the development of this thesis are also enumerated.

Chapter 2

Related Work

2.1 Introduction

Recently, Multiple Sequence Alignment (MSA) is the most important task in bioinformatics. MSA consists of aligning more than two input DNA, RNA or protein sequences at a time, allowing the biologist to identify similarity or divergence regions with the objective of knowing relevant biological information about the compared sequences. This biological information is useful for identifying evolutionary or functional relationships between sequences and predicting sequence structures [Not02].

In this chapter, the main works in the scientific community related to the MSA are collected and analyzed. Section 2.2 briefly introduces Pairwise Sequence Alignments methodologies. Section 2.3 describes the most important MSA heuristics and some of the most popular methods. Section 2.4 studies the most important MSA benchmarking databases to validate and compare the MSA methods and evaluate the alignment accuracy. Finally, Section 2.5 provides an overview of the application of High Performance Computing (HPC) techniques to MSA in order to solve the MSA complexity problem and improve its performance. This section describes some techniques that inspired us in the development of some methods proposed during this thesis.

2.2 Pairwise Sequence Alignment Methods

As explained in previous sections, a Pairwise Sequence Alignment consists of aligning two sequences. Before introducing the main MSA heuristics and tools, it is necessary to review briefly the most common Pairwise Sequence Alignment techniques. Next, three Pairwise Sequence Alignment methods are described, with special emphasis on Dynamic programming due to its importance in many MSA techniques.

2.2.1 Dot-matrix Methods

A dot-matrix plot is a graphic method that allows the comparison of two biological sequences and identifies regions of close similarity between them [Mou07]. Dot plots, like the one shown in Figure 2.1, are used as a technique for displaying information that permits certain sequences features, such as insertions, deletions, repeats, to be identified visually.

To construct a dot-matrix plot, one of the sequences is written along the x-axis, and the other along the y-axis of a two-dimensional matrix. Then a dot is placed at any point where the characters in the appropriate columns match. Once the dots have been plotted, these are combined to form lines. The dot-matrix plot shows, as a single line along the main diagonal of the matrix, the more similar related sequences.

The main problems of dot plots are noise, lack of clarity and non-intuitiveness; making it difficult to identify the properties of the sequences. Furthermore, these methods are limited to two sequences due to them being time-consuming.

2.2.2 Dynamic Programming

Dynamic programming is an alignment technique to identify the globally computation optimal alignment solution [Den03]. It can be applied to produce both global and local alignments implementing the Needleman-Wunsch (N-W) or Smith-Waterman (S-W) algorithms respectively [NW70][SW81].

Both algorithms work similarly, building a two-dimensional matrix where

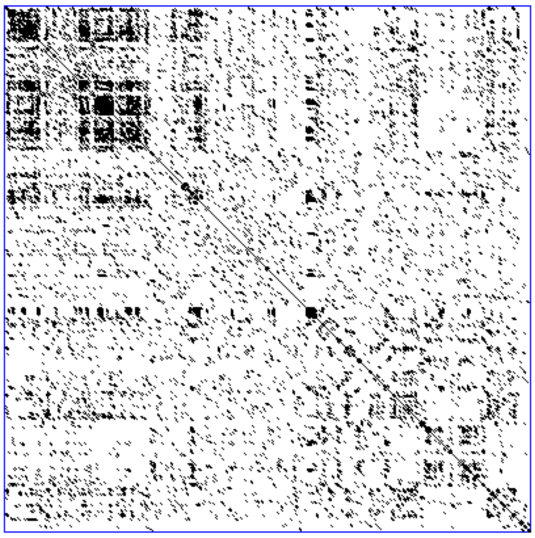


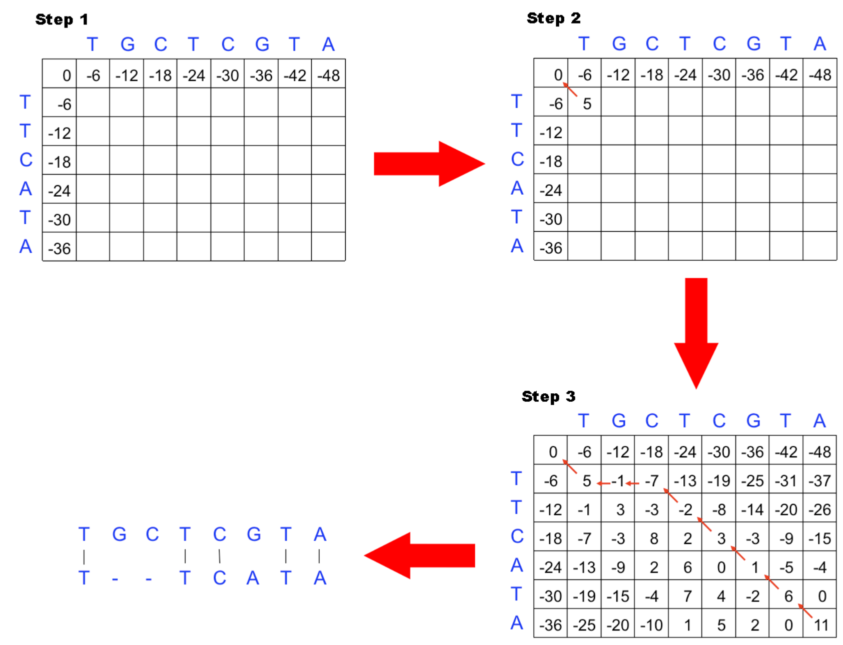
Figure 2.1: Example of a DNA dot plot of a human zinc finger transcription factor

the sequence residues are written on the x-axis and the other sequence residues on the y-axis. The two algorithms can be divided into the following three steps, which can be seen as examples in Figure 2.2a for N-W algorithm and Figure 2.2b for S-W algorithm.

1. Initialize the matrix. This consists of filling the first row and first column of the matrix, which they do not correspond to any residue of the two sequences. For N-W, the first row and the first column of the score matrix are filled as multiples of the opening gap penalty, while for S-W, this row and column are filled with 0.

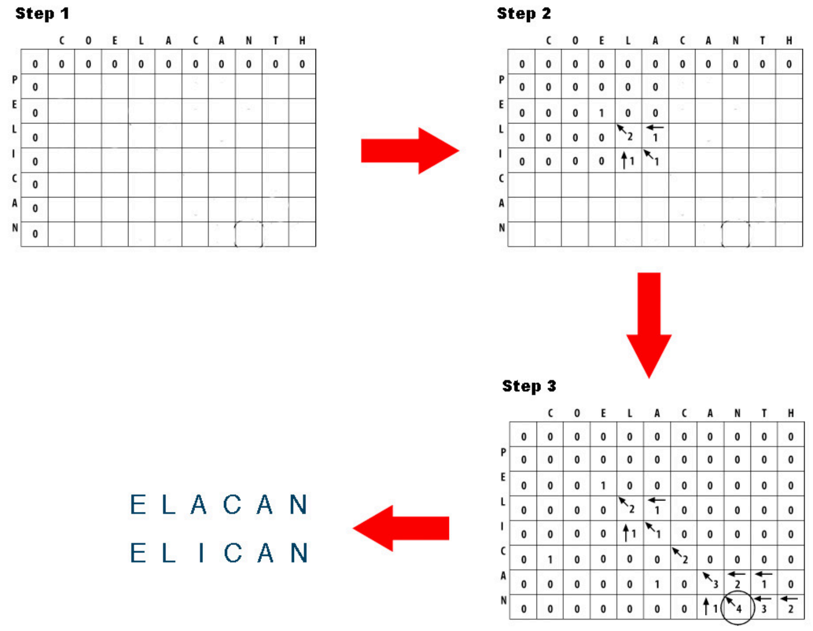
2. Fill the matrix cells from the top-left corner with the scores and gap penalties to compare the matches or mismatches between the residues of the two sequences.

   For protein alignments, a substitution matrix is used to assign scores to residue matches or mismatches. A protein substitution matrix is a symmetrical $20x20$ matrix based on any property of amino acids [DS78]. The most important matrices are evolutionary matrices, like the PAM [Alt91] and BLOSUM [HH92] families. Both matrix families are derived from the analysis of known alignments of closely related sequences, although



		T	G	C	T	C	G	T	A
	0	-6	-12	-18	-24	-30	-36	-42	-48
T	-6	5	-1	-7	-13	-19	-25	-31	-37
T	-12	-1	3	-3	-2	-8	-14	-20	-26
C	-18	-7	-3	8	2	3	-3	-9	-15
A	-24	-13	-9	2	6	0	1	-5	-4
T	-30	-19	-15	-4	7	4	-2	6	0
A	-36	-25	-20	-10	1	5	2	0	11

(a) Neddleman-Wunsch algorithm example



	C	O	E	L	A	C	A	N	T	H
0	0	0	0	0	0	0	0	0	0	0
P 0	0	0	0	0	0	0	0	0	0	0
E 0	0	0	1	0	0	0	0	0	0	0
L 0	0	0	0	2	1	0	0	0	0	0
I 0	0	0	0	1	1	0	0	0	0	0
C 0	1	0	0	0	0	2	0	0	0	0
A 0	0	0	0	0	1	0	3	2	1	0
N 0	0	0	0	0	0	0	1	4	3	2

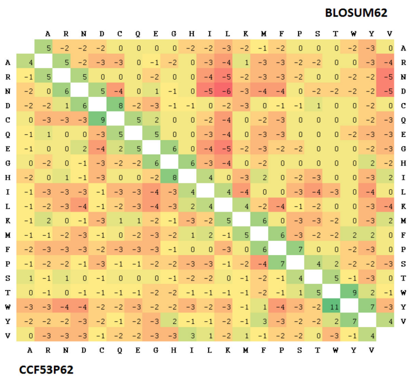
(b) Smith-Waterman algorithm example

Figure 2.2: Dynamic Programming examples

BLOSUM matrices are newer and considered better. Figures 2.3a and 2.3b show examples of a BLOSUM62 and PAM250 matrix respectively. On the other hand, for DNA and RNA alignments, an identity matrix, which is a more simple scoring scheme, is used. This scheme usually consists of assigning a positive match score, a negative mismatch and a negative gap penalty.

For gap penalties, a common extension to standard linear gap costs is the use of two different gap penalties; for opening and for extending [VW94].

In this step, the difference between the N-W and S-W algorithms is that in the S-W all the negative scores in the matrix are set to 0.

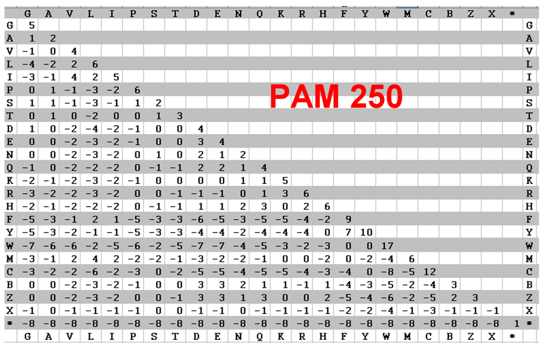


BLOSUM62

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	
		5	-2	-2	-2	0	0	0	0	-2	-2	-3	-2	-1	-2	0	0	0	-2	-3	0	A
A	4		5	-2	-3	-3	0	-1	-2	0	-3	-4	1	-3	-3	-2	-2	0	0	-3	-4	R
R	-1	5		5	0	0	0	-2	0	0	-4	-5	-2	-3	-3	-2	0	0	-2	-2	-5	N
N	-2	0	6		5	-4	0	1	-1	0	-5	-6	-3	-4	-4	0	-2	-2	-2	-2	-5	D
D	-2	-2	1	6		8	-2	-3	-1	-1	0	-2	-3	0	-1	-1	1	0	0	-2	0	C
C	0	-3	-3	-3	9		5	2	0	0	-2	-4	0	-2	-3	0	0	0	0	-2	-3	Q
Q	-1	1	0	0	-3	5		5	0	0	-3	-4	0	-3	-3	0	0	0	-2	-3	-3	E
E	-1	0	0	2	-4	2	5		6	0	-4	-5	-2	-3	-2	-2	0	0	0	-2	-3	G
G	0	-2	0	-1	-3	-2	-2	6		6	-3	-4	0	-2	0	0	0	0	0	2	-2	H
H	-2	0	1	-1	-3	0	0	-2	8		4	0	-3	2	0	-2	-3	0	0	-3	2	I
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4		4	-4	0	0	-3	-4	-3	0	-4	0	L
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4		4	-2	-4	-1	-2	0	0	-3	-4	K
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5		6	0	-3	-3	-2	0	-3	2	M
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5		6	-3	-2	-2	2	2	0	F
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6		7	0	0	-2	-3	0	P
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7		4	2	-2	-2	-3	S
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4		5	-1	-3	0	T
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5		9	2	-1	W
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11		7	-3	Y
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7		4	V
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4		
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V		

CCF53P62

(a) BLOSUM62 substitution matrix



PAM 250

	G	A	V	L	I	P	S	T	D	E	N	Q	K	R	H	F	Y	W	M	C	B	Z	X	*	
G	5																								G
A	1	2																							A
V	-1	0	4																						V
L	-4	-2	2	6																					L
I	-3	-1	4	2	5																				I
P	0	1	-1	-3	-2	6																			P
S	1	1	-1	-3	-1	1	2																		S
T	0	1	0	-2	0	0	1	3																	T
D	1	0	-2	-4	-2	-1	0	0	4																D
E	0	0	-2	-3	-2	-1	0	0	3	4															E
N	0	0	-2	-3	-2	0	1	0	2	1	2														N
Q	-1	0	-2	-2	-2	0	-1	-1	2	2	1	4													Q
K	-2	-1	-2	-3	-2	-1	0	0	0	0	1	1	5												K
R	-3	-2	-2	-3	-2	0	0	-1	-1	-1	0	1	3	6											R
H	-2	-1	-2	-2	-2	0	-1	-1	1	1	2	3	0	2	6										H
F	-5	-3	-1	2	1	-5	-3	-3	-6	-5	-3	-5	-5	-4	-2	9									F
Y	-5	-3	-2	-1	-1	-5	-3	-3	-4	-4	-2	-4	-4	-4	0	7	10								Y
W	-7	-6	-6	-2	-5	-6	-2	-5	-7	-7	-4	-5	-3	-2	-3	0	0	17							W
M	-3	-1	2	4	2	-2	-2	-1	-3	-2	-2	-1	0	0	-2	0	-2	-4	6						M
C	-3	-2	-2	-6	-2	-3	0	-2	-5	-5	-4	-5	-5	-4	-3	-4	0	-8	-5	12					C
B	0	0	-2	-3	-2	-1	0	0	3	3	2	1	1	-1	1	-4	-3	-5	-2	-4	3				B
Z	0	0	-2	-3	-2	0	0	-1	3	3	1	3	0	0	2	-5	-4	-6	-2	-5	2	3			Z
X	-1	0	-1	-1	-1	-1	0	0	-1	-1	0	-1	-1	-1	-1	-2	-2	-4	-1	-3	-1	-1	-1		X
*	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	-8	1	*
	G	A	V	L	I	P	S	T	D	E	N	Q	K	R	H	F	Y	W	M	C	B	Z	X	*	

(b) PAM250 substitution matrix

Figure 2.3: Substitution matrices examples

3. Produce the alignment from the matrix through a traceback process that allows three movements: diagonally, up or left. This process consists of maximizing identity by maximizing an objective function. The objective function is usually the Weighted Sum-of-Pairs [AL89].

   For the N-W algorithm, the traceback process starts at the bottom-right corner of the matrix until it reaches the top-left corner. On the other hand, the S-W algorithm starts from the maximum score and follows it to a score of zero, aligning only a region of the sequences and discarding the rest. In both algorithms the sequences are aligned backwards.

As mentioned above, dynamic programming guarantees finding the computational optimal alignment given a particular objective function. However, the mathematically perfect objective function does not guarantee that this will also be the biologically optimal.

2.2.3 Word Methods

Word methods, also called k-tuple methods, are heuristics used to identify a series of short and non-overlapping subsequences, which are called words, in a sequence, which are then matched to other sequences. The relative positions of the word in the two compared sequences are subtracted to obtain an offset value. If distinct words produce the same offset, this will be an alignment region. If this region is detected, these methods apply more sensitive alignment criteria. If not, these unnecessary comparisons with sequences with no appreciable similarity are eliminated.

These methods do not guarantee that an optimal alignment solution will be found, but are significantly more efficient than dynamic programming. They are useful in large-scale database searches and are implemented in such popular database search tools as the FASTA and BLAST families [PL88][AGM+90].

2.3 Multiple Sequence Alignment Methods

MSAs are computationally more complex than Pairwise Sequence Alignments, therefore MSAs require more sophisticated methodologies. Nowadays, most common MSA programs use heuristics rather than global optimization owing to the identification of the optimal alignment between more than a few sequences being computationally almost impossible.

Over recent years, hundreds of methods for producing MSAs have been published based on different heuristics. In this section, the philosophy of some of these heuristics and methods are briefly explained and they are divided into different categories. However, many of the following methods include characteristics of more than one category. First, global optimization or exact methods and their constraints are presented. Then the most important heuristics proposed in the literature are explained.

2.3.1 Exact Algorithms

Exact or global optimization algorithms are high-quality heuristics that are able to build alignments very close to the optimal. The main problem of these heuristics is their complexity, as they only allow a small number of sequences to be aligned and are also limited to the Sum-of-Pairs objective function [WJ94]. Three different exact algorithm techniques are described below.

- **Dynamic programming:** The dynamic programming solution to Pairwise Alignment can be extended to align more than two sequences. In this case, to align N sequences, the method must construct an N-dimensional matrix increasing the search space exponentially by increasing the N and the length (L) of these sequences. In summary, the complexity of DP is $O(L^N)$, being a NP-complete problem that limits the method to align just only three sequences [WJ94][Jus01].

- **MSA program:** This is a heuristic implementation of Carrillo and Lipman's algorithm [CL88] for aligning protein, DNA and RNA sequences [LAK89]. It attempts to produce an optimal MSA using a branch and bound technique. The major idea is to identify in advance the portion of the hyperspace that does not contribute to the solution and exclude it from the computation. It minimizes the sum of the pairwise costs, weighting the pairs using information derived from an evolutionary tree. Therefore, MSA program reduces the computational demands of dynamic programming, making it possible to align up to ten sequences. Despite these improvements, MSA program is also limited in the number of sequences it can align because of its high memory and time requirements. Furthermore, it is not guaranteed to find the mathematically optimal alignment.

- **DCA:** This is a program for producing MSA of proteins, RNA, or DNA sequences based on the Divide-and-Conquer algorithm [SMD97]. DCA cuts the sequences into subsets of small segments. Then, these segments are aligned using the MSA program. Finally, the resulting sub-alignments are reassembled by DCA. To avoid losing too much accuracy, the major trick is to cut the sequences at the right positions in order to produce an alignment which remains as close as possible to the optimal. Despite the improvements achieved by the application of the Divide-and-Conquer algorithm to reduce the complexity problems, DCA using the MSA program is still limited by the memory and time requirements, it being unable to align many more sequences (20-30 sequences).

2.3.2 Progressive Alignment

The progressive alignment technique proposed by Hogeweg [HH84] and redefined by Feng and Taylor [FD87][Tay88] is nowadays the most widely used heuristic. It is based on the successive construction of pair-wise alignments, building the alignment progressively. It starts by aligning the two most closely related sequences, traditionally using dynamic programming, and then adds sequences in order of increasing distance. The order for selecting the sequences to be added is determined by a guide tree previously obtained from an initial pair-wise alignment.

This heuristic has a great advantage of speed and simplicity combined with reasonable sensitivity. On the other hand, the major disadvantage of progressive methods is that they are very dependent on the initial alignments and thus more likely to perform well for closely related sequences. Errors made during any stage in the growth of MSA are then propagated to the final alignment. For these reasons, progressive alignment heuristic is not guaranteed to be globally optimal.

The most popular progressive alignment algorithm implementation is the Clustal family [HS88], especially the weighted variant ClustalW which is provided by a large number of web portals [THG94]. Then there is ClustalX, a variation of the Clustal family that implements a graphic user interface [TGP$^+$97]. Other progressive alignment implementations are MULTALIGN [Cor88] and Kalign [LS05]. Finally, T-Coffee [NHH00] is another progressive alignment implementation combined with a consistency-based scheme which is more accurate, but slower, than ClustalW.

To explain the progressive alignment algorithm in greater detail and because it has been used during this work, the main steps in ClustalW are described next.

ClustalW

ClustalW, from the Clustal family and proposed by Thompson in 1994 [THG94], is the most widely used alignment method. The ClustalW alignment process requires the three following main stages to produce the progressive MSA (Figure 2.4):

1. The distance matrix is constructed from the relationships between sequences. A distance matrix is an NxN matrix, N being the number of sequences, and it represents the pairwise distances between sequences in sequence space. Initially, the pairwise distances between sequences were obtained by doing all-against-all pairwise alignments between all sequences, but nowadays, faster statistical algorithms can be used to obtain these distance values, such as the KTUP distances.

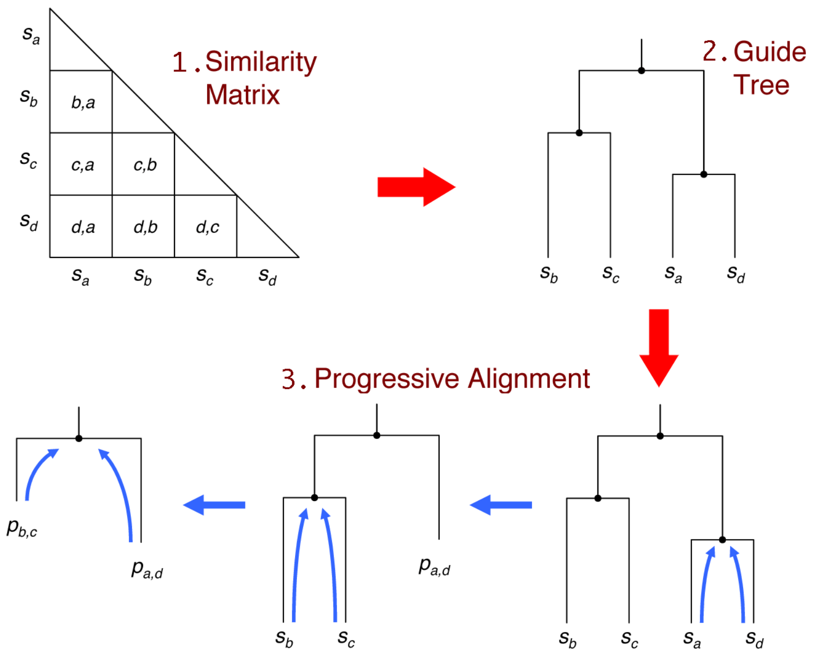


Figure 2.4: Progressive Alignment algorithm

2. The guide tree from the relationships between sequences defined by the distance matrix is produced. The guide tree is built by a clustering method to calculate distance trees, such as neighbor-joining [SN87] or UPGMA [SM58], using the pairwise distances extracted from the distance matrix.

3. The MSA is built by aligning the sequences sequentially to the growing MSA according to the order defined by the guide tree.

2.3.3 Iterative Alignment

As explained above, progressive methods improve efficiency at the cost of accuracy. Iterative algorithms, proposed by Barton [BS87], attempt to improve the heavy dependence on the accuracy of the initial pair-wise alignments exhibited by progressive alignment. They are based on the idea that an accurate alignment of a given set of sequences can be computed by modifying a preexisting non-optimal alignment. In the algorithm proposed by Barton, an initial alignment is generated then one sequence is taken out and realigned to the remaining sequences. The process is repeated until all the sequences have been realigned.

Although iterative methods generally give more accurate alignments than progressive methods, the major disadvantages of iterative MSAs are inherited from the optimization methods: the process can get trapped in a local minimum and the improvement in the alignment accuracy comes at the expense of a longer run time.

Iterative methods can be classified into stochastic or non-stochastic iterative algorithms. In the former, the modifications are done using random protocols, while in the latter, non-stochastic, ones these are done using dynamic programming. Some algorithms of each category are reviewed below.

2.3.3.1 Stochastic Iterative Algorithms

Stochastic algorithms attempt to refine alignments modifying them randomly and trying to maximize an objective function. These methods try to model the random behavior of evolutionary changes in sequences, such as mutations.

Simulated Annealing, Genetic Algorithm and Hidden Markov Models are representative stochastic iterative techniques. Next, these algorithms are briefly described.

Simulated Annealing

The Simulated Annealing technique (SA) [MRR+53] consists of refining an existing MSA, produced by another method, and maximizing an objective function as a Sum-of-Pairs. This process can be divided into four steps:

1. An initial sub-optimal alignment is generated using an MSA method.
2. The initial alignment is randomly modified rearranging the alignment.
3. The modified alignment is scored using an objective function.
4. The modified alignment is kept or discarded depending on a metaphorical "temperature factor" that determines the rate at which rearrangements happen and the likelihood of each rearrangement. Typical usage alternates between periods of high rearrangement rates with relatively low likelihood of exploring more distant regions of alignment space, with periods of lower rates and higher likelihoods of exploring local minimums near the new regions.
5. The process continues until convergence is met.

The main disadvantage of SA is that it is too slow at producing alignments and therefore can only be used to improve the accuracy of existing alignments.

This approach has been implemented in Multiple Sequence Alignment using Simulated Annealing (MSASA) [KPC94].

Genetic Algorithms

The Genetic Algorithms (GA) [Hol92][Gol89] for MSA are a much faster alternative to SA. The idea is to use genetic algorithms to produce a refined MSA simulating the evolutionary process.

GAs are heuristics based on the principles of natural evolution and survival of the fittest described in Darwin's theory of evolution. Solutions to the problem are represented by an encoded string, analogous to chromosomes in genetics. The algorithm starts with a number of random generated solutions, the so-called population. Then a series of genetic operators (selection, crossover, mutation and replacement) are applied to the solutions in the population to produce a new population. It is based on the principle of the survival of the fittest, which consists of measuring the fitness of each solution using an objective function (fitness function), before the fittest solutions are chosen in the selection stage to contribute to new solutions. New solutions are formed by the crossover operation, where two of the chosen solutions are selected and recombined to form new solutions. A small proportion of these new solutions are then mutated, i.e. changed slightly in a random way. Once an appropriate number of new solutions has been created, these replace an equivalent number of old solutions, with some of the fitter old solutions surviving to the next generation. The process is continued for a number of generations until a solution optimizing the objective function is found.

GAs for MSA are very useful as research tools, but their main disadvantage is that they are too slow for large-scale projects or every-day use.

A popular application that uses a classic GA to refine an alignment is SAGA [NH96]. SAGA randomly generates multiple MSAs of a given set of sequences and evolves them under some selection pressure. Within SAGA, fitness depends on maximizing an objective function, originally the Sum-of-Pairs score. Therefore, alignments will die or survive over the generations depending on their fitness. As stated before, alignments can also improve and reproduce through genetic operators known as mutations and crossovers. In SAGA, mutations randomly insert or shift gaps, while crossovers combine the content of two alignments. The complete disconnection between the operators and the original objective function made it possible to modify the original fitness function Sum-of-Pairs score to COFFEE score [NHH98]. While SAGA is used for protein sequences, an equivalent to SAGA, called RAGA [NOH97], was designed for RNA sequences.

Hidden Markov Models

Hidden Markov Models (HMM) are probabilistic models that assign likelihoods to all possible combinations of gaps, matches and mismatches to determine the most likely MSA or set of possible MSAs [KBM$^+$94][Edd95].

An HMM is a statistical Markov model, proposed by Andrey Markov [Mar71], in which the system being modeled is assumed to be a Markov process with unobserved or hidden states [BP66]. A Markov process is a stochastic process where future states are only dependent on the choice of the current state, not on the sequence of events that preceded it. HMM can be seen as a finite state machine that moves through a series of states and produces some kind of output, either when the machine has reached a particular state or when it is moving from one state to another. The HMM generates a protein sequence by emitting amino acids as it progresses through a series of states. Each state has a table of amino acid emission probabilities. There are also transition probabilities for moving from state to state.

Figure 2.5 shows the most popular topology in sequence analysis for a HMM. Note that there are three kinds of states represented by three different shapes. The squares are called match states, and the amino acids emitted from them form the conserved primary structure of a protein. These amino acids are the same as those in the common ancestor or, if not, are the result of substitutions. The diamond shapes are inserted states and emit amino acids that come from insertions. The circles are special silent states known as delete states and model deletions. Transitions from state to state progress from left to right through the model, with the exception of the self-loops in the diamond insertion states. The self-loops allow deletions of any length to fit the model, regardless of the length of other sequences in the family.

Any sequence can be represented by a path in the model. The probability of any sequence is computed by multiplying the emission and transition probabilities along the path.

An efficient search variant of the dynamic programming method, known as the Viterbi algorithm [Vit67], is generally used to align the growing MSA successively to the next sequence in the query set to produce a new MSA. The Viterbi model tries to first to find the most probable path. The probability of

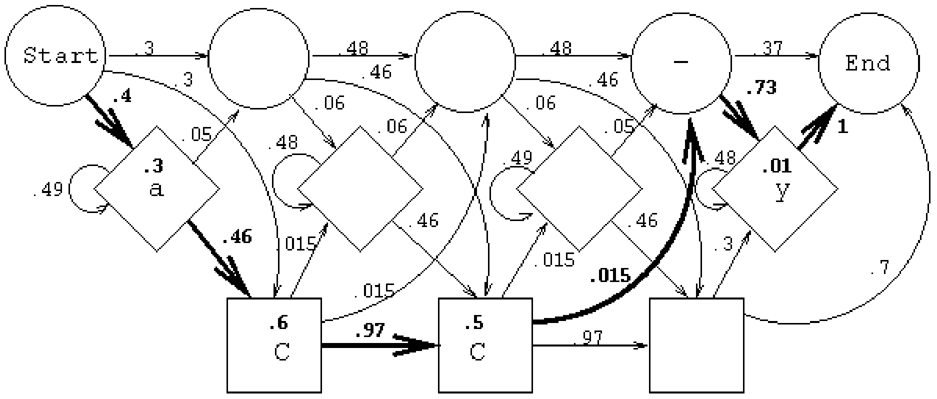


Figure 2.5: A possible HMM model for the protein ACCY

the sequence obtained from the HMM model is then computed by multiplying all probabilities along the path. What the algorithm basically does is that, at every stage in its journey, it tries to figure out and select the most probable path leading from one particular amino acid emission to another, after having compared the probability scores corresponding to each path. The algorithm does the above every time it wants to go from one amino acid to another. The resulting path that stretches through the whole sequence of amino acids is said to be the most probable path. However, like progressive alignment methods, this technique can be influenced by the order in which the sequences are integrated into the alignment, especially when the sequences are distantly related.

HMMs are used to create both global and local MSAs from a set of unaligned sequences and to classify sequences. The most popular HMM implementations are POA [GL04], SAM [HK96] and HMMER [DEKM98].

2.3.3.2 Non-stochastic Iterative Algorithms

Non-stochastic iterative algorithms work similarly to progressive methods but they consist of correcting the alignment later by repeatedly re-aligning an initial MSA using standard dynamic programming. The MSA is re-iterated, starting with the pair-wise re-alignment of sequences within subgroups, called profiles, and then the re-alignment of these profiles. The procedure finishes when iterations fail to improve the alignment.

An important step that affects the variability of the method is the way in which sequences are divided into groups before being re-aligned. The choice of subgroups can be made via sequence relations on the guide tree, random selection, etc, depending on the implementation.

The most common non-stochastic iterative methods are Muscle [Rob04], PRRN/PRRP [Got96], MAFFT [KMKM02] and ClustalΩ [SWD$^+$11]. To understand what a non-stochastic iterative alignment is and how it works, the process and the main stages used by MUSCLE to produce a refined alignment are explained below.

Muscle

MUSCLE, proposed by Edgar in 2004 [Rob04], is an application for creating multiple alignments of protein sequences. The basic strategy used by MUSCLE is similar to that used by PRRP and MAFFT. A progressive alignment is built, to which horizontal refinement is then applied. As shown in Figure 2.6, Muscle can be divided into three stages. On completion of each stage, a multiple alignment is available and the algorithm can be terminated.

1. The goal of the first stage is to produce an MSA, emphasizing speed over accuracy. This is achieved by producing the progressive alignment MSA using the kmer distances [Edg04]. This measure does not require an alignment, giving a significant speed advantage. This stage is divided into three more steps:

   (1) The kmer distances are computed for each pair of input sequences, giving a distance matrix.

   (2) A distance matrix is clustered by UPGMA, producing a binary tree.

   (3) A progressive alignment is constructed by following the branching order of tree.

2. The second stage consists of improving the first MSA, correcting the errors produced by the approximate kmer distance measure. MUSCLE therefore re-estimates the tree using the Kimura distance [Kim80], which

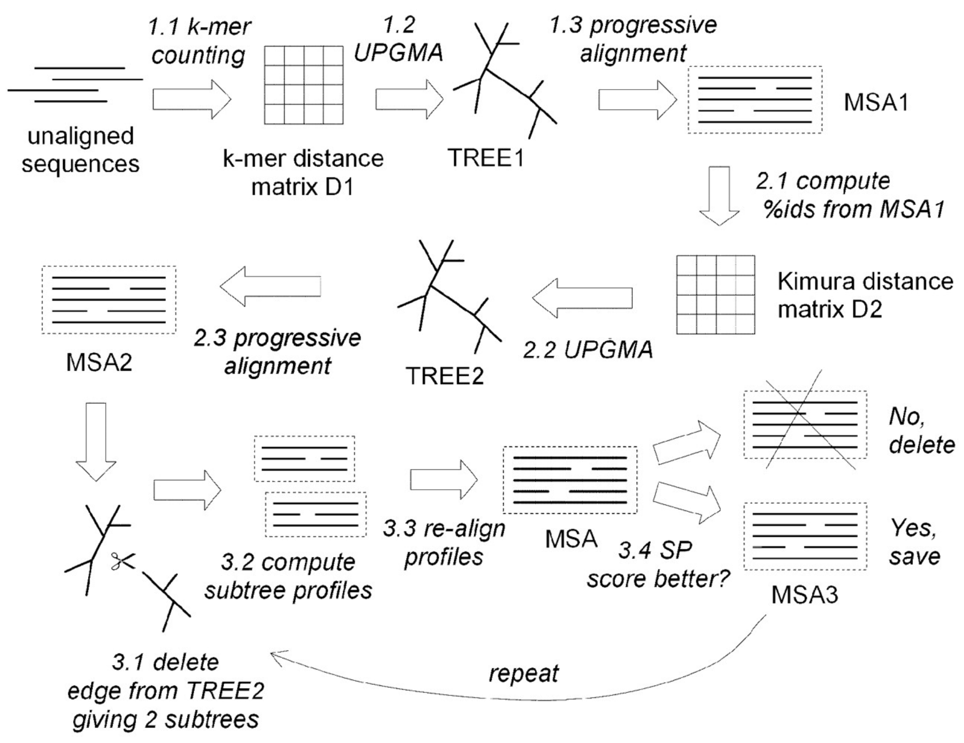

1.1 k-mer counting
1.2 UPGMA
1.3 progressive alignment
unaligned sequences
k-mer distance matrix D1
TREE1
MSA1
2.1 compute %ids from MSA1
Kimura distance matrix D2
2.2 UPGMA
TREE2
2.3 progressive alignment
MSA2
3.1 delete edge from TREE2 giving 2 subtrees
3.2 compute subtree profiles
3.3 re-align profiles
MSA
3.4 SP score better?
No, delete
Yes, save
MSA3
repeat

Figure 2.6: Muscle algorithm

is more accurate, but requires an alignment. This process consists of the following three steps:

(1) The Kimura distances for each pair of input sequences are computed from the MSA obtained in stage 1, giving another distance matrix.

(2) This new matrix is clustered by UPGMA, producing a binary tree.

(3) A progressive alignment is produced following the tree order, thus generating an optimized MSA.

3. The third stage is a refinement process, where the tree obtained in the second stage is modified to optimize the final MSA. The Muscle refinement process consists of the following steps:

   (1) An edge of the tree is chosen and then this tree is divided into two subtrees by deleting the edge.

   (2) Each subtree is aligned producing two profile alignments.

(3) A new MSA is produced by re-aligning the two profiles.

(4) This new MSA is evaluated using the Sum-of-Pairs score.

(5) If the SP score has improved, the new alignment is kept, otherwise it is discarded.

(6) This refinement process is repeated until convergence or until a user-defined limit is reached.

To sum up, MUSCLE demonstrates improvements in accuracy and reductions in computational complexity by exploiting a range of existing and new algorithmic techniques.

2.3.4 Consistency-based Methods

Consistency-based methods were designed to overcome the accuracy limits caused by the greediness problems of progressive and iterative aligners. As explained above, this problem consists of mistakes made early in the alignment process being moved to the final alignment and which can be difficult to correct using iterative methods. The solution proposed by the consistency-based approach, which was originally described by Gotoh in 1990 [Got90] and later redefined by Vingron and Argos in 1991 [VA91], is to introduce the available information about the sequences and use this to avoid mistakes in the alignment. The ideal situation is to use simultaneously all the information in the sequences that lets these mistakes be avoided easily, but this goal is computationally unrealistic. The method must incorporate the necessary information at a computational cost.

The common idea of consistency-based approaches is to evaluate pairwise alignments though the comparison of third sequences. This consistency information can then be used to construct the alignments or evaluate them, depending on the approach. The first combination of a consistency-based scoring scheme with a progressive alignment algorithm was described by Notredame in T-Coffee [NHH00]. Other common MSA programs that use consistency to produce accurate alignments are Probcons [DMBB05], Probalign [RULD06], and Dialign [MFDW98].

Next, because T-Coffee has been the reference MSA application to study during this work, it is introduced and its main stages described in order to understand how a consistency-based method works.

T-Coffee

T-Coffee (TC), proposed by Notredame in 2000 [NHH00], is a multiple sequence aligner method that combines the consistency-based scoring function COFFEE [NHH98] with the progressive alignment algorithm. T-Coffee provides an improvement in accuracy compared over most methods based on a progressive strategy, as errors made in the initial alignments cannot be rectified later as the rest of the sequences are added in. In contrast, T-Coffee introduces a library generated using a mixture of pair-wise alignments in order to reduce greediness and increase accuracy. However, the introduction of these improvements has penalized T-Coffee in speed as compared to the most commonly used alternatives. As can be seen in Figure 2.7, T-Coffee is divided into three main stages:

1. **Primary Library**. The primary library contains a set of pairwise alignments from among all the sequences to be aligned. Originally, the library was generated by combining the ten top-scoring non-intersecting local alignments constructed with Lalign program [HM91] and all the global pair-wise alignments obtained with ClustalW [THG94]. However, in the current standard versions, the library is built from all against all pairwise alignments computed with a pair of Hidden Markov Models algorithms. In the library, each alignment is represented as a list of pairwise residue matches. A sequence identity weight is assigned to each pair of aligned residues in order to reflect the correctness of a constraint. This stage is the most time and memory consuming, limiting its applicability to no more than 200 sequences on a typical workstation.

2. **Extended Library**. The extended library allows the TC to reduce errors made in the initial alignments. The extension of the library is a re-weighting process where the new weights for a given pair of sequences also depend on information from the other sequences in the set. Originally,

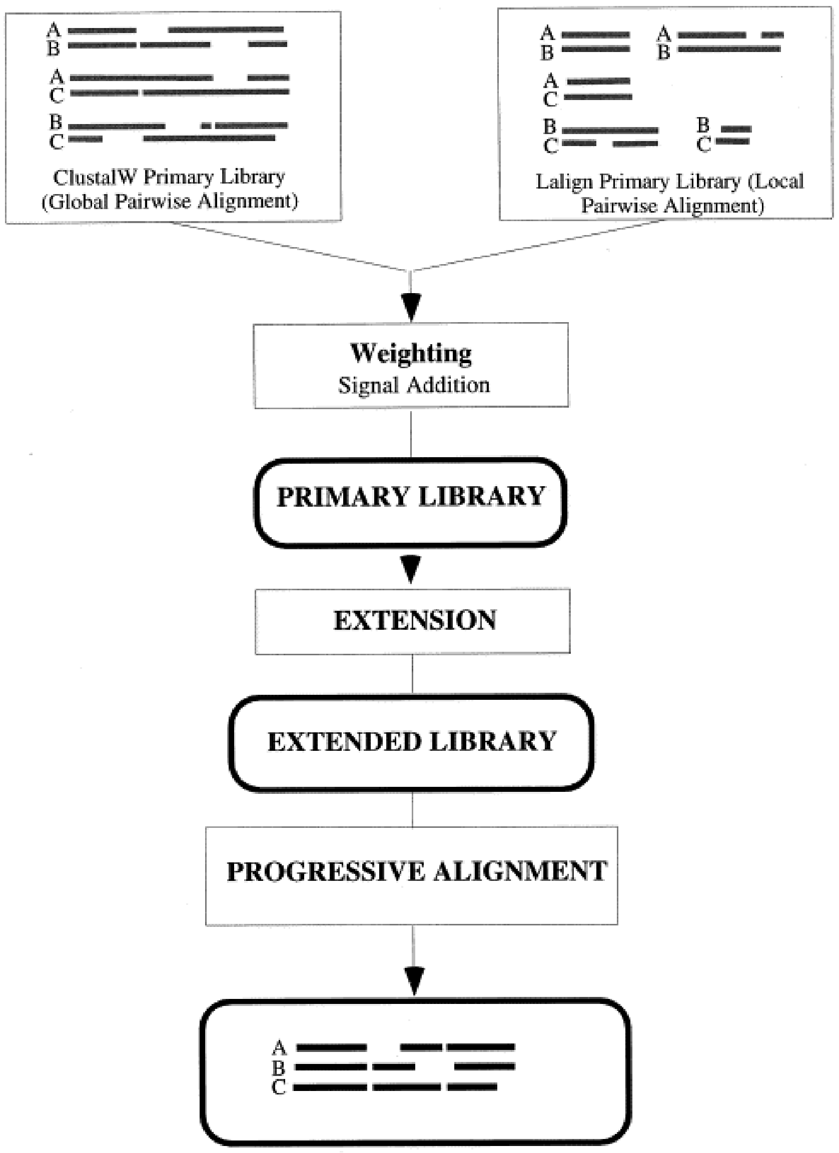

ClustalW Primary Library (Global Pairwise Alignment)
Lalign Primary Library (Local Pairwise Alignment)
Weighting
Signal Addition
PRIMARY LIBRARY
EXTENSION
EXTENDED LIBRARY
PROGRESSIVE ALIGNMENT

Figure 2.7: T-Coffee algorithm

the library extension was an independent process, but in last versions, the TC was improved by doing the extension online during the progressive alignment stage using only the related sequences.

3. **Progressive Alignment strategy**. The MSA is produced by the progressive alignment strategy explained in Section 2.3.2. First of all, all-against-all pairwise alignments are made to construct a distance matrix between all the sequences. The distance matrix is then used to generate the guide tree. Finally, the sequences are aligned progressively by following the order of the guide tree. The main difference is that the alignments are done with the dynamic programming technique and maximizing the COFFEE objective function using the weights in the extended library instead of using the substitution matrix weights and gap penalties.

2.3.5 Meta-aligners

The huge number of MSA methods and the lack of a globally accepted solution make it harder for biologist to choose a specific method. It is known that there is a close dependency between the phylogenetic and structure modeling in the chosen aligner [WSH08]. For example, phylogenetic trees may vary significantly depending on the applications used to produce the alignment. Furthermore, benchmarks also demonstrate that no method outperforms all the others, and that it is almost impossible to predict which method will outperform all the others on a specific dataset with enough certainty. For example, one method can outperform all the others in a specific dataset. However, this may not happen in other datasets. Without enough structural data or functional information, it is hard to compare different alignments, which are obtained from the available methods, and select the best resulting alignment.

Meta-aligners are an attempt to address this issue. The main idea is to use the output of some available MSA methods as information for computing a consistent alignment that may be considered as some sort of the average of all the alignments considered.

M-Coffee [WOHN06], from the T-Coffee package, was the first method designed to be used as a meta-method. M-Coffee consists of computing alternative MSAs from a given set of sequences using any selected method. By default, M-Coffee combines eight of the most accurate and distinct MSA packages. Each of the resulting alignments is then turned into a primary library and all of them are merged into the main T-Coffee library. The resulting library is used to compute an MSA consistent with the original alignments.

2.3.6 Template-based Methods

Template-based MSA methods consist of using the information of a template to enrich a sequence [Tay86]. This template information can be used to guide the sequence alignment in a sequence independent fashion. A template can either be a 3D structure, a profile or a prediction of some kind.

The use of structural or profile information increases the accuracy of the resulting alignments. Depending on the nature of the template, one refers to its usage as structural extension or homology extension.

Figure 2.8 shows the use of three possible types of templates (homology extension, structure and functional annotation) in T-Coffee package. Templates are compared with a suitable method and the resulting alignment is mapped onto the final alignment of the original target sequences. The residue pairs thus identified are then incorporated into the primary library.

Structural extension

Given two sequences with a homolog in PDB, which is a structures database [BWF$^+$00], structural extension consists of superposing the PDB structures accurately and mapping the resulting alignment onto the original sequences. This protocol produces an alignment with all the properties of a structure-based sequence alignment. Generally, structures are better conserved than sequences, so the addition of structural information should provide a more biologically significant MSA.

Structural extension only defines pairwise alignments, but was initially implemented in 3D−Coffee [OSA$^+$04] for protein alignments. Later, it was implemented in several RNA MSA methods as T-Lara [BKR05], MARNA [SB05] and R-Coffee [WHN08] using RNA secondary structures as templates.

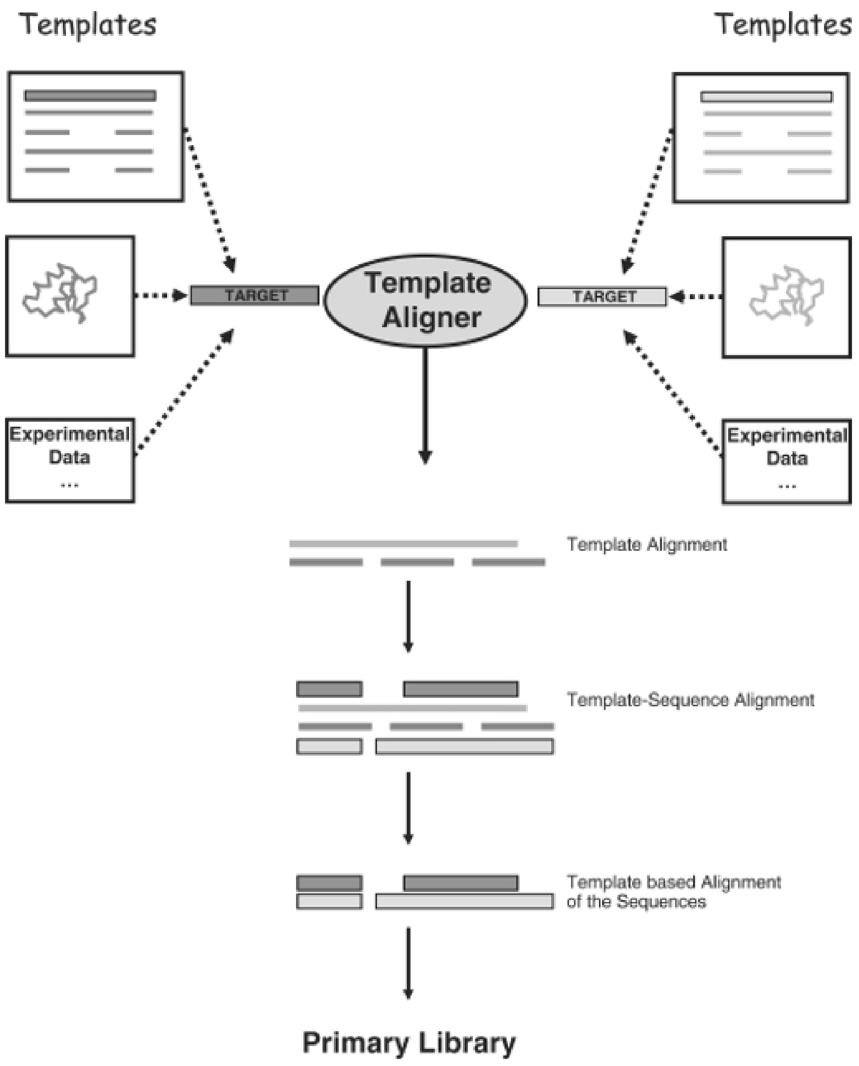


Figure 2.8: Template-based protocols in a T-Coffee primary library

In the T-Coffee case, templates are used to accurately align the sequences taking into account the predicted structures and then the resulting structure-based pairwise alignments are combined into the T-Coffee primary library, as explained above.

Homology extension

Homology extension works similarly to structural extension but uses profiles rather than structures. In this case, sequences are replaced with profiles containing homologs. Profiles are later used to progressively generate the MSA. Although profiles could be built using any available technique, fast methods like PSI-BLAST, [AGM$^+$90] have been favored.

The purpose of introducing profile information is to obtain highly accurate alignments, although the use of structural extension is the best way.

The first homology extension protocol was described in the PRALINE package [SH05]. Then, another popular homology extension methods like PROMALS [PG07] and PSI-Coffee of T-Coffee package were implemented. Both methods the sequences are associated to PSI-Blast profiles.

2.4 Benchmarking

Due to the fierce competition to become the most accurate MSA method, there are nowadays multiple structure-based reference alignment databases to evaluate alignments and validate or compare new MSA methods. An alignment produced by an MSA application is compared with the corresponding reference alignment, giving an accuracy score. These reference alignments have usually been constructed using 3D protein structural alignments, and are thus independent of sequence alignment methods. The quality of the MSA programs has typically been assessed by an accuracy score that measures the proportion of correctly aligned residue pairs or alignment positions in the alignment.

A comprehensive evaluation and comparison of alignment programs requires a large number of accurate reference alignments that can be used as test cases. McClure in 1994 [MVF94] was the first to evaluate her alignments by assessing the correct alignment of pre-defined functional motifs. She demonstrated that the performance of alignments programs depends on the number and length of the sequences, the degree of similarity between sequences, the number of insertions in the alignment, the existence of large insertions, N/C-terminal extensions and over-representation of some members of the protein family. Benchmarking datasets have to take these characteristics into account to evaluate MSA programs correctly.

The use of MSA reference alignment collections for benchmarking is very convenient because of its simplicity. However, a major problem is the heavy reliance on the correctness of the reference alignment.

Next, the most common used databases are described, with greater emphasis placed on the ones used in this work.

- **BAliBASE:** BAliBASE was defined by Thompson in 1999 [TPP99] and is a database of high-quality documented and manually refined reference alignments based on 3D structural superpositions to identify the strong and weak points of the alignment programs. In BAliBASE, the alignments are categorized by core blocks of conservation sequence length, similarity, and the presence of insertions and N/C-terminal extensions.

  Originally, BAliBASE consisted of 142 reference alignments, but in 2005 it was updated to 217 reference alignments to include new, more challenging test cases, representing the real problems encountered when aligning large sets of complex sequences.

  The accuracy of the alignments is measured with *bali_score*, an application supplied with the database. It compares the user alignment against a reference alignment and it returns two standard accuracy measures: the Sum-of-Pairs score (SP) and the Total Column score (TCS). The SP score is used to determine the extent to which the program succeeds in aligning some of the sequences in an alignment. This score increases with the number of sequences correctly aligned. The TCS score is a binary score that tests the ability of the programs to align all the sequences correctly.

- **PREFAB:** PREFAB was created by Edgar in 2004 [Rob04] and it is the main alternative to BAliBASE. Prefab is a very extensive collection of 1,682 pairs of homologous structures gathered by PSI-BLAST. Prefab test cases are generated by taking a pairwise alignment of sequences of known 3D structures, and adding up to 24 high scoring homologues for each sequence.

One of the main differences is that PREFAB is not an MSA collection like BAliBASE since each dataset only contains one pair of structures. Therefore, PREFAB is less stringent than BAliBASE, where accuracy can be tested on entire multiple alignments columns rather than pairs of residues.

PREFAB uses three accuracy measures: Quality score (Q), Total Column score (TCS) and APDB. Q score is the number of correctly aligned residue pairs divided by the number of residue pairs in the reference alignment, while TCS is the number of correctly aligned columns divided by the number of columns in the reference alignment. TCS is the same as BAliBASE TCS and it is equivalent to Q in the case of two sequences. Finally, APDB is derived from the structures alone and no reference alignment of the sequences or structures is required.

- **HOMSTRAD:** This is a protein alignment database from sets of sequences, published by Mizuguchi in 1998 [MDBO98], in which all the members have a known 3D structure. HOMSTRAD contains 130 protein families and 590 aligned structures, which were selected on the basis of the quality of the X-ray analysis and accuracy of the structure. It was not specifically designed as a benchmark database, although it is regularly employed as such.

- **OXBench:** This is a data set of reference alignments and software tools for benchmarking pairwise and multiple alignment methods, developed by Raghava in 2003 [RSA$^+$03]. The benchmark data set is made up of domain families obtained from the 3D database of protein structural domains and it comprises three related datasets: The MASTER set test cases that deal with isolated domains derived exclusively from sequences with a known structure. Then the FULL set was generated from suitable

MASTER test cases, using full-length sequence data. Finally, the EXTENDED set in which high scoring homologous sequences were added to each MASTER test case to generate it.

- **SABmark:** This is a sequence alignment benchmark published by Van-Walle in 2005 [WLW05] that provides sets of multiple alignment problems derived from the SCOP classification.

  It is divided into two subsets. Each test group in the SUPERFAMILY set represents a SCOP superfamily, whose sequences are 25–50% identical. Each test group in the TWILIGHT set represents a common SCOP fold and sequences are 0–25% identical. In addition, these two subsets are also provided with non-homologous (false positive) sequences included in each group. Instead of a single alignment acting as a reference, SABmark provides multiple pairwise references for each test, and it is the average score from each of these references that is taken here as a score for each test case.

- **IRMBASE:** IRMBASE was published by Subramanian in 2005 [SMKM05]. It is entirely different to the other benchmarks, because its test cases contain a number of simulated motifs [SEM98] inserted into otherwise random (unalignable) sequences. The test cases are designed to examine whether a method is able to detect isolated motifs within sequences, and so are tailored to a local alignment approach.

2.5 High Performance Computing in MSA

It is known that the computational cost of an optimal sequence alignment is in the order of O(L^N) given L, the length of each sequence, and N, the number of sequences. This means that the exact solution to the problem is intractable and it is only possible to align few sequences. However, although heuristic approaches reach a sub-optimal solution in a reasonable time, nowadays large data sets demand faster and more efficient algorithms.

Recently, several types of parallel algorithms have been implemented in different parallel systems to address these computationally intensive problems.

Some of these algorithms are focused on solving dynamic programming computational problems and the others are focused on parallelizing the whole MSA heuristic.

2.5.1 Parallel Algorithms for Pairwise Alignment

The first parallel approximations were applied to the Pairwise Alignment, in an attempt to parallelize the Needleman-Wunsch and Smith-Waterman dynamic programming algorithms in order to align more than two sequences.

However, the main problem of these algorithms is the data dependencies derived from the similarity matrix. Therefore, most of the parallel strategies proposed in the literature by Chen [CS03], Driga [DLS$^+$06] and Batista [BM06] are based on the use of the wave-front method, since the similarity matrix calculations, which can be done in parallel, evolve as waves on matrix diagonals (see Figure 2.9).

2.5.1.1 Wavefront Approach

In the wavefront approaches, the dynamic programming matrix is evenly partitioned into rectangular regions. Each region is assigned to a unique processor. However, when this ends the region computation, it can process others. The parallel processing starts with one processor computing the entries of the top-left tile, meanwhile the remaining processors are idle. Next, all the region with its up and right neighbor entries available can be calculated. In this sense, in the first iteration region numbered with a "1" can be processed. In the second iteration, two regions numbered with a "2" will be calculated, and so on.

As it can be see in figure 2.9, each diagonal of tiles labelled with the same number forms a wavefront line. The parallelism degree depends on wavefront size. At the P-step, all the P processors can work in parallel because the wavefront line consists of exactly P regions.

2.5.1.2 Divide and Conquer Approach

Other approaches, like the one proposed by Rajko [RA04], use a divide-and-conquer paradigm to distribute the calculation of the similarity matrix evenly

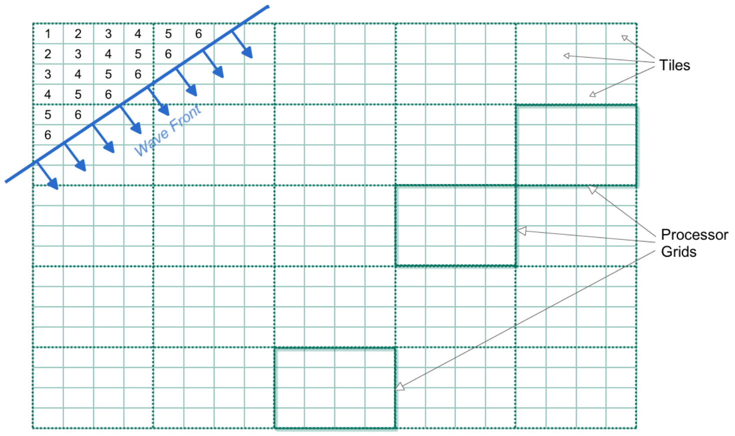


Figure 2.9: Wavefront parallelism in Dynamic Programming alignments

among the processors. Another approach commonly used by other MSA methods is the Myers-Miller dynamic programming algorithm [MM88], which solves the pairwise alignment problem by dividing the matrix into half and then scanning from opposite corners towards the middle. This point becomes the corner of two sub-blocks, which in turn are divided and scanned for midpoints. The recursion continues until a trivial alignment is encountered. The forward and backward scans can occur briefly in parallel, but they must join before determining the midpoint (Figure 2.10).

However, in these parallelization methods, scalability is limited by the length of the sequences. Very long sequences are required in order to take advantage of the high number of processors. All these parallel strategies are implemented and tested in the literature for CPU, GPU or FPGA computing systems [SM11][BLB09].

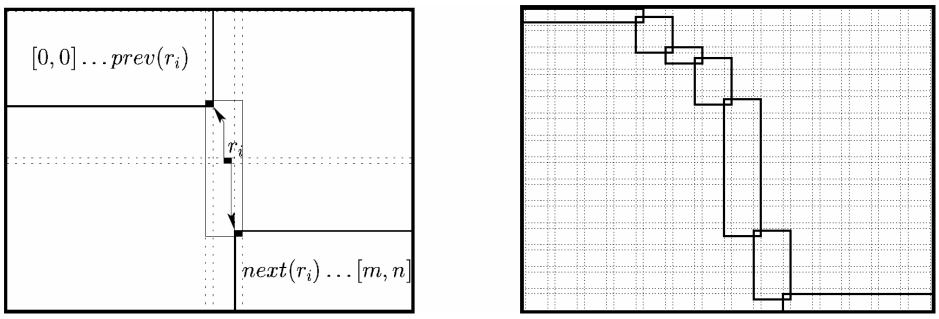


Figure 2.10: Divide and Conquer approach: first and final divisions.

2.5.2 Parallel Multiple Sequence Alignments

Furthermore, there are other approaches designed to improve the alignment performance by parallelizing the whole MSA application. There are three main approaches to increasing the performance of MSA tools: the shared-memory parallel versions based on multi-threads or multi-processes, parallelization based on the distributed memory paradigm that uses MPI, and finally the ones, like GPUs and Hadoop, that uses the new HPC architectures.

2.5.2.1 Multi-Threading and Multi-Process Parallel Aligners

Nowadays, many methods are being updated to be executed concurrently to take advantage of multi-core machines, such as implementing each computational execution as an operating system process, or implementing the computational processes as a set of threads within a single operating system process.

Multi-Threading ClustalW

The easiest way to paralellize an application is to use the shared-memory paradigm. It is for this reason, that firsts approaches for improving the aligner performance are based on implementing the multi-threading or multi-process version of these tools.

MT-ClustalW [CKT06] tool is the multi-threading version of ClustalW. In this version, the neighbor-joining guide tree is built using multiple threads. Therefore, each thread calculates a portion of the neighbor-joining distance matrix. In the same way, all pairwise alignments and the progressive alignment are calculated.

Although MT-ClustalW is able to improve the execution time of ClustalW, its scalability is very limited by the algorithm and computational resources available on the computer.

Multi-Threading MAFFT

MAFFT [KMKM02] is another aligner that has opted to parallelize its code using the multi-threading technique. All its three calculation stages, namely the all-to-all comparison, the progressive alignment and iterative refinement, from the MAFFT MSA program were parallelized using the POSIX Threads library.

- **All-to-all comparison.** Multiple threads process different pairwise alignments simultaneously and independently, with little loss of CPU time.
- **The progressive alignment.** In this stage, group-to-group alignment calculations are performed along with a guide tree. This process is not very suitable for parallelization, because the order of the alignment calculations is restricted by the guide tree. That is, an alignment at a node cannot be performed until all of the alignments in its child nodes have been completed. Thus, the efficiency of this stage is very limited due to these data dependencies.
- **Iterative refinement process.** In each step of the refinement, an alignment is divided into two sub-alignments and then, the two sub-alignments are realigned to obtain an alignment with a higher score. A simple hill-climbing approach is implemented to assign random realignments to multiple threads and perform them in parallel. When a new alignment obtains a better score than the original one, it replaces the original.

Cloud-Coffee

Cloud-Coffee [TOG+10] is a multi-process version of the popular T-Coffee package. Cloud-Coffee executes the following stages concurrently:

- **Template selection**. T-Coffee automatically associates structural templates with user-provided sequences. Templates can either be identified by running a BLAST against an appropriate database or produced by modeling (i.e. RNA secondary structure prediction). This simple, but computationally intensive, process of iterating on the list of sequences is parallelized by Cloud-Coffee.

- **Library computation**. Library computation involves computing pair-wise (or multiple) alignments using some pre-defined method. In this phase, parallelization is achieved by grouping the alignment tasks into a number of individual jobs equal to the number of available processors. Jobs are then submitted simultaneously, and their output merged into a single library by the master process.

- **Library extension**. The library extension involves re-evaluating the score for aligning every residue pair. This operation is achieved by considering every pair of sequences in turn and subsequently updating the library. It can therefore be parallelized using the library computation strategy.

- **Progressive alignment**. The progressive alignment stage involves aligning sequences (or profiles) two by two, while following the order indicated by a binary guide tree. Parallelization is achieved by processing all the independent nodes separately, with optimal speedup achieved when dealing with perfectly balanced trees.

2.5.2.2 Distributed Memory Parallel Aligners

When using distributed memory HPC architectures, the most popular approaches are also implemented in the MPI library to be executed on clusters of workstations. Thus, the main MSA methods have their own parallel versions, like ClustalW-MPI [Li03], Parallel-TCoffee [ZYRA07], Dialign-P [SNKM04] and Muscle [Rob04]. All of them share a similar parallel design based on the division of the multiple alignment into three main stages that can be processed separately:

1. Calculation of the distance matrix required to construct the guide tree, with a complexity of O(N^2L^2), where N is the number of sequences and L the typical sequence length.

2. Generation of the guide tree based on the neighbor-joining or UPGMA clustering methods, with a complexity of O(N^3).

3. A Progressive alignment stage where all nodes in the guide tree without dependencies are aligned in parallel, with a complexity of $\mathrm{O}(N^3 + NL^2)$.

The first and second stages, distance-matrix calculation and guide tree generation, correspond to an allocating problem where the solution corresponds to determining the best match between the time-independent tasks and the computing nodes. Thus, these steps can be implemented by a master-worker paradigm using fixed-size chunking [KW85] or guided self-scheduling strategies [PK87]. The efficiency of the last stage, the parallel progressive alignment, obviously depends on the topology of the tree. The problem with this method is that an unbalanced tree can severely limit the number of parallel tasks. For a well-balanced guide tree, the ideal speed-up can be estimated as $N/\log N$, where N is the number of nodes in the tree. However, even a balanced tree will have limited parallelism near the root and guide trees are usually unbalanced.

ClustalW-MPI

ClustalW-MPI [Li03] is the distributed-memory parallel version of the popular aligner. The parallelization of ClustalW-MPI follows the general scheme presented in the previous section. The distance-matrix calculation is parallelized as independent tasks, achieving a consistent linear speed-up. The guide tree generation has been optimized to decrease its complexity from $O(N^3)$ to $O(N^2)$. Finally, the progressive alignment stage is parallelized using a mixed fine-grain and coarse-grain approach. In the coarse-grain, all vertices in the guide tree without data-dependences are aligned in parallel. The fine-grain parallelism is applied to the profile-to-profile alignment, using the recursive parallelism paradigm.

The performance and scalability of ClustalW-MPI is optimal in the first two steps, but very limited in the progressive alignment stage. The impact of this bottleneck on global performance can be a very limiting factor for large-scale alignments.

Dialign-P

Dialign is a versatile tool for pair-wise and multiple alignment of nucleic acid and proteins sequences [MFDW98]. The Dialign sequential algorithm performs all respective optimal pair-wise alignments in order to build up a multiple alignment in greedy fashion. This implies that, for a set of N input sequences, $N * (N-1)/2$ pair-wise alignments must be calculated.

The parallel version, Dialign-P [SNKM04], takes advantage that these pair-wise alignments are completely independent of each other. Thus, they can be calculated in parallel, dividing the run time by the number of parallel processors. The most critical point in this approach is how to distribute the workload evenly to the processors. The authors have opted to use a task mapping greedy algorithm. This algorithm sorts the pairwise alignment tasks by their expected run time (an estimated based on the sequence lengths) and then assigns the tasks to the processors in a round-robin fashion, starting with the longest ones.

Parallel-TCoffee

Parallel-TCoffee (PTC) [ZYRA07] is a parallel version of T-Coffee (TC) that allows the reporting of alignments of more than hundreds of sequences, which is far beyond the capability of the sequential version. PTC is implemented on version 3.79 of TC and supports most of the options provided by this.

The implementation of PTC uses a distributed master-worker architecture and a message-passing paradigm employing one-sided communication primitives. Basically, PTC parallelizes the library generation, the progressive alignment, which are the two main and most difficult stages of TC, and the distance matrix computation.

- **Distance Matrix Computation**. In TC, the progressive alignment strategy is guided by a neighbor-joining tree. This is generated using some measure of sequence similarity expressed with a distance matrix. The computation of the distance matrix requires $\binom{n}{2}$ sequence comparisons and each comparison is a totally independent task. This is why PTC parallelizes it through a master-worker paradigm and implements a Guided Self Scheduling method (GSS) [PK87] to distribute the computations (tasks) among workers. Each worker computes its part of the distance matrix, calculates the time required and returns the results to the master.

- **Library Generation**. The library generation consists of three phases:

  1. Generation of all pair-wise constraints. The method implemented by PTC to parallelize this part is similar to the distance matrix computation. PTC uses a modified GSS, where half of the total number of pairwise alignments are distributed proportionally based on worker efficiency, and the other part is distributed using GSS. Finally, each worker stores the list of the corresponding constraints in its local memory instead of returning the results to the master. The efficiency of the workers is known due to the time it takes each processor to compute its part of the distance matrix.
  2. Deletion, association and re-weighting of duplicated pair-wise constraints. Each host merges its duplicate constraints locally using the original TC method, and then PTC implements a parallel sorting to group and merge all the repeated constraints that are found by different workers.
  3. Transformation of the library into a three-dimensional look-up table. The library is turned into a three-dimensional look-up table, where the rows are indexed by sequences and the columns indexed by residues. PTC implements the table using one-sided remote memory access mechanisms (RMA). Each worker creates a read-only RMA window that presents its part of the table, and all the workers share two indexing vectors to retrieve any address of any entry in the table. Finally, each worker also implements a cache system managed by a Last Recently Used policy to store all the frequent requests to the remote memory.

- **Progressive Alignment**. The computations of the progressive alignment stage follow a tree order, and their parallelization can be reduced to a Directed Acyclic Graph (DAG) scheduling problem. This is why this stage is the most difficult to parallelize.

PTC implements a strategy similar to the HLFET (Highest Level First with Estimated Times) algorithms [KA99]. It launches the graph nodes that have no precedence dependencies and allow the earliest start time, until all graph nodes (alignments) are computed.

2.5.3 GPUs based Parallel Aligners

More recently, different approaches use graphics processing units (GPUs) to reduce the execution time of MSA applications [Jun09]. The development of GPU-ClustalW [LSVMW06], MSA-CUDA [LSM09] or MUMmerGPU [TS09] are examples of such applications.

GPU-ClustalW

The main advantage of GPUs compared to other accelerator architectures, such as FPGAs, is that they are commodity components. In particular, most users already have access to PCs with modern graphics cards. However, it is necessary to reformulate the alignment algorithms and data structures using computer graphics primitives (e.g. triangles, textures, vertices, fragments) in order to facilitate efficient use of these resources. Furthermore, restrictions of the underlying streaming architecture have to be taken into account, i.e. random access writes to memory are not supported and no cross fragment data or persistent state is possible.

GPU-ClustalW [LSVMW06] uses graphics processing units (GPUs) as a computational platform to accelerate MSAs with ClustalW. As the distance matrix calculation is the most expensive stage of ClustalW algorithm (consuming 90% of the execution time) the authors opted to reformulate this stage in terms of computer graphics primitives. In this sense, they parallelize the Smith-Waterman algorithm, taking advantage of the fact that all elements in the same anti-diagonal of the distance matrix can be computed independently in parallel.

Although this solution achieves a speedup of 7, its scalability is very limited by the sequence lengths. They cannot take full advantage of the GPU architecture due to the data dependencies of the Smith-Waterman algorithm.

MSA-CUDA

The MSA-CUDA [LSM09] is a parallel MSA program which parallelizes the three stages of ClustalW, processing a pipeline using CUDA. This implementation achieves significant speedups compared to the sequential version.

The first stage is parallelized considering the pairwise distance computation of one pair of sequences as a task. Then each task is assigned to exactly one thread and dimBlock tasks are performed in parallel by different threads within the thread block.

To parallelize the neighbor-joining tree, all the tree nodes are stored in a vector and the relationship between nodes is maintained through vector indices instead of pointers. Each node object stores the indices of itself, its parent and its left and right children, and accesses them using the vector index. One thread block is assigned to compute the difference value of the means of the branch lengths on the left and right of one node, which is selected as the reference. Every thread in the thread block is assigned to perform the computation on a separate sub-set of leaf nodes. For each leaf node in a subset, the corresponding thread identifies on which side of the selected node this leaf node lies and computes the distance between this leaf node and the selected node.

In MSA-CUDA, the progressive alignment is conducted iteratively in a multi-pass way. For each pass, firstly, all undone alignments that can be performed in this pass are identified by checking the flag words of their left and right children stored in the flag-vector. If both of their left and right children have been aligned, this alignment is added to the ready alignment list managing all the alignments to be performed in this pass. Otherwise, this alignment has to wait until both of its children have been aligned. After the completion of the ready alignment list, the pairs of profiles corresponding to those alignments are constructed. Secondly, the pairwise alignments of all pairs of profiles are performed on the GPU in parallel. Thirdly, gaps are added to the sequences corresponding to each pair of profiles by tracing back to its optimal alignment. Finally, all the alignments performed in this pass will set their flag words in the flag-vector to indicate that they are aligned.

The MSA-CUDA improves the performance of the previous GPUs versions of ClustalW. It achieves similar speedups as the ClustalW-MPI executed in a cluster with 32 CPU-node but using only a GPU (nVIDIA GeForce GTX 280 graphics card).

2.5.4 Conclusions

In the spite of the parallelization of the most popular aligners, using different HPC paradigms and architectures its scalability and performance is very limited. Nowadays, large-scale alignments with thousands of sequences is still a big challenge.

The main difficulties for its parallelization are the high data dependencies that constrain the number of tasks that can be executed in parallel. Moreover, as some methods, such as consistency-based ones like T-Coffee [NHH00] and Dialing [MFDW98], are more time consuming than alternative applications, the main challenge for their parallelization is the memory required to build the consistency data structures. Therefore, the main efforts in the parallelization of consistency-based methods are guided towards a shared-memory paradigm. For example, Cloud-Coffee [TOG$^+$10] is a new multi-process version of the T-Coffee package or Parallel-TCoffee [ZYRA07], which is able to aggregate the CPU and memory of a cluster of workstations using an RMA (Remote Memory Access) mechanism to share the library between different processors. However, neither the shared-memory nor message-passing approaches solve the bottleneck that the library represents for the scalability of the consistency-based methods.

Other key challenge in the parallelization of MSA tools is to maintain the quality of the original methods. Performance cannot be changed by accuracy. Therefore, take into consideration the continuous improvement of these packages, the parallel version needs to be integrated inside the original version in order to take advantage of further improvements in the basis alignment algorithms.

Finally, is also important to provide easy access to the parallel software and HPC infrastructure in order for biologist to be able to use these new aligners. Sometimes the simplicity of utilization of a tool is more important than accuracy or performance when users choose an aligner. In this sense, such new HPC infrastructures as Cloud can be a determinant factor in the popularization of parallel aligners.

Chapter 3

Balanced Guide Tree

3.1 Introduction

In spite of the improvement in speed introduced by heuristics, the computational requirements for large-scale alignments (thousands of sequences) clearly exceed the workstation performance. Therefore, parallel implementations based on the main heuristics, such as ClustalW-MPI, Parallel-TCoffee or DialignP, were implemented. Although, as explained in Section 2.5.4, all of these improve their original algorithm, these distributed methods exhibit scalability problems when the number of sequences increases, as they are constrained by data dependencies that guide the alignment process.

This chapter introduces a new clustering algorithm, applied to progressive alignment heuristics, in order to exploit the degree of parallelism in the final alignment process. This goal is achieved by building a guide tree that increases the number of available parallel alignments in order to take advantage of increasing computer resources. To evaluate it, the algorithm presented was implemented in both T-Coffee (TC) and its parallel version Parallel-TCoffee (PTC). The first was used to test the performance in a multi-core workstation, and the second one in a distributed system, such as a cluster.

This chapter is organized as follows: Section 3.2 introduces and analyzes the scalability problems of parallel implementations. Section 3.3 is devoted to the presentation of the proposed algorithm, its variations and its main features. In Section 3.4, experimentation is performed to evaluate the effectiveness of the proposed algorithm. Finally, Section 3.5 summarizes the main conclusions drawn from the experimental results.

3.2 Problem analysis

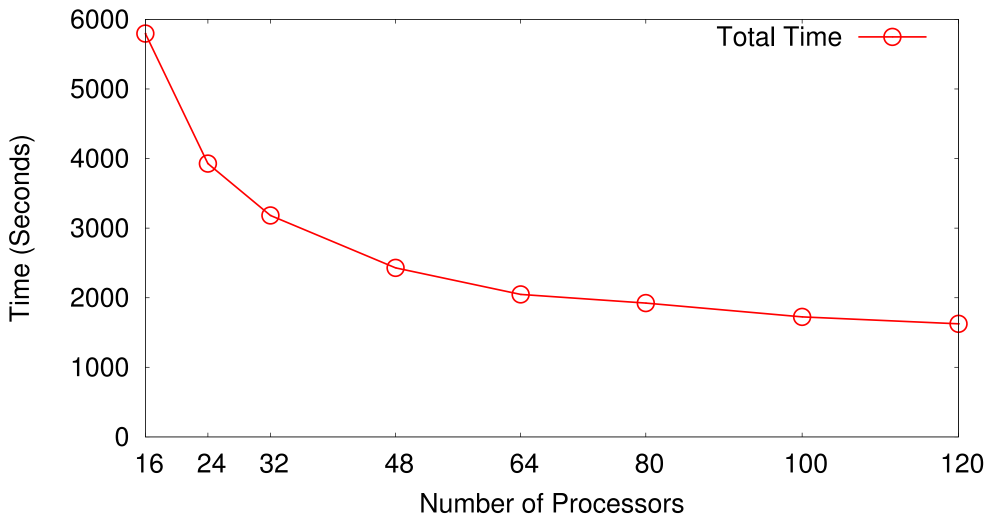
It is known that one of the problems of MSA parallel implementations is that these methods do not scale when the number of sequences increases. Before treating the problem, it is necessary to analyze these applications to find the root of the problem.

This section is devoted to analyzing the scalability problems of parallel progressive alignment methods when the number of sequences increases. This study was done using PTC, the MPI version of TC and described in Section 2.5.2.2. Although it implements an old version of TC and thus does not obtain current accuracy results, PTC allows the behaviour of the scalability in larger sets of sequences to be visualized. However, the conclusions drawn from this analysis can be extended to concurrent TC.

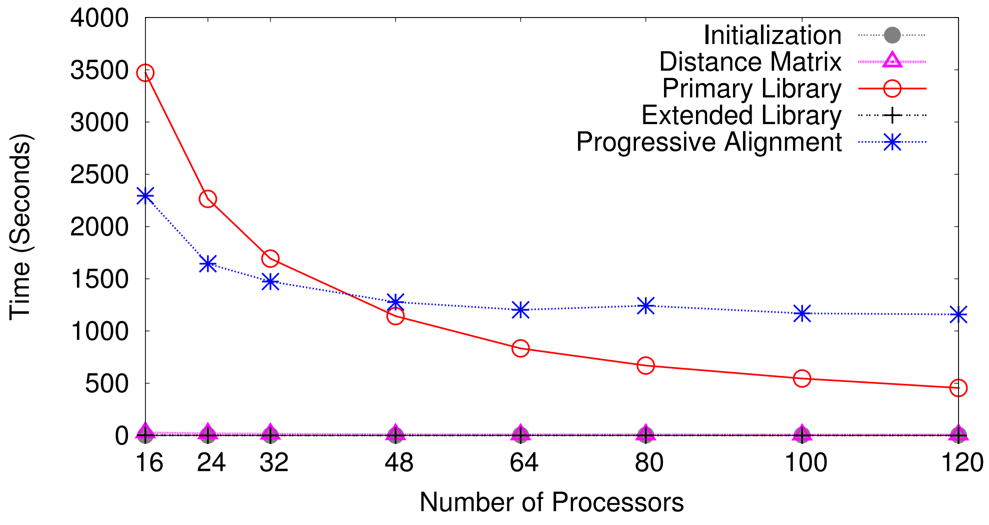
To analyze the scalability of PTC, some tests were carried out varying the numbers of processors and using different sequence sets from the Pfam database [SED97]. In the present manuscript, the results from the PF00231 dataset are shown in Figures 3.1a and 3.1b. PF00231 is made up of 554 sequences and a maximum length of 331 amino-acids. This experiment was run on a Cluster using from 16 to 120 processors.

Figure 3.1a shows the time PTC required to perform the alignment as a function of the number of processors used to execute the parallel alignment. PTC improves the execution time as the number of processors increases. As we can see, up to 64 processors, the speedup is nearly optimal, but with more than 64 processors, the speedup stopped increasing stuck, limiting the scalability of PTC.

Figure 3.1b shows the same study but breaking down the execution time of PTC in the five different stages of TC algorithm: Initialization, Distance Matrix, Primary Library calculation, the extension of the consistency library and finally, the Progressive Alignment. At first glance, we can notice that the most time-consuming steps are the calculation of the Primary Library and the



(a) PTC total execution time



(b) PTC stages execution time

Figure 3.1: PTC scalability analysis

Progressive Alignment. These two stages consume more than the 98% of the execution time. Figure 3.1b also shows that the only stage that is improved is the library generation while the progressive alignment stage remains linear with the number of processors, diminishing the scalability. That is because the generation of the primary library can be divided into several completely independent tasks, which is an ideal situation to be implemented by the master-worker paradigm. Meanwhile, it is known that progressive alignment stage is bounded by task dependencies extracted from the guide tree.

Given these results, we deduce that optimization must be focused on the progressive alignment stage.

3.2.1 Progressive Alignment analysis

As explained above, this stage is driven by the neighbor-joining guide tree (NJ) that determines the order of the partial alignments in the progressive alignment. Therefore, the guide tree is the key that defines the dependences among parallel tasks.

Figure 3.2 displays, as an example, a guide tree representation generated by the NJ algorithm, implemented in both TC and PTC. The PT-nodes (internal nodes) define the progressive alignment tasks. The leaf nodes are the different sequences to align (12 in this example) and the tree represents the order in which such progressive partial alignments can be performed. From the point of view of parallelism, only PT-nodes with all dependencies resolved (all children nodes are leaves) can be executed as independent tasks. The example has three initial tasks, grey PT-nodes, which can be executed in parallel. This value defines the maximum number of tasks that can be launched in parallel. Another important parameter in order for modeling the parallelization of progressive alignment is the critical path, which is the longest path through the guide tree to obtain the final alignment. The critical path defines the number of sequential iterations that the algorithm has to perform. The more sequential iterations there are, the lower parallelism, the lower the performance and the higher the execution times. In the example, the maximum degree of parallelism is 3 parallel tasks and the length of the critical path is 7 iterations.

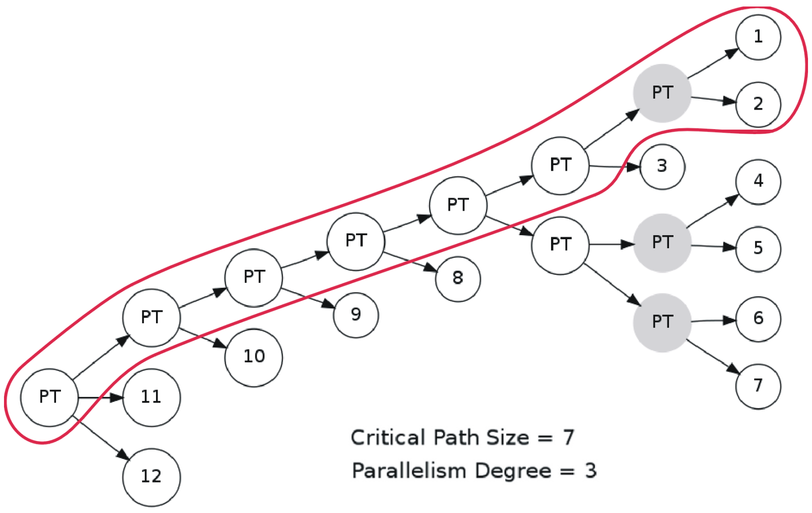


Figure 3.2: Guide tree generated with standard NJ heuristic

From this tree example, it can be deduced that the shortest critical path and the larger degree of parallelism is achieved by a perfectly balanced tree. In Table 3.1, this study is extended to different NJ guide trees obtained from the PF00074, PF00200, PF00231, PF00349, PF01057 and PF08443 sequence sets from the Pfam database. This study estimates the number of sequences, the critical path (CP), the optimal critical path (OCP), (obtained from the perfectly balanced tree), the maximum degree of parallelism (MPD), and the optimal maximum parallelism degree (OMPD). The optimal critical path can be estimated as the $OCP = \log_2 N - 1 + 1$, N being the number of sequences to align. The optimal maximum parallelism degree is achieved in the first iteration and can be estimated as $OMPD = 2^{(OPC-2)}$. These optimal values can only be achieved when the tree is perfectly balanced, i.e. all the levels are complete.

From this table, it can be deduced that the standard NJ algorithm builds guide trees with long critical paths, compared to the optimum values (OCP), and small degrees of parallelism if they are compared to the optimal maximum degree of parallelism (OMPD). For instance, for the PF01057 sequence set, the critical path length is 94 sequential iterations, while the optimal is only 10, and the maximum parallelism degree is 71, while the optimal is 281. This behavior

		NJ Guide Trees			
Sequences	**N seqs**	**CP**	**OCP**	**MPD**	**OMPD**
PF00074	442	28	9	107	220
PF00349	515	23	10	143	257
PF00231	554	26	10	164	276
PF01057	563	94	10	71	281
PF00200	594	29	10	173	296
PF08443	749	68	10	215	374

Table 3.1: NJ guide tree features for some sequence sets from the Pfam database

can be seen in the remainder of the tests. Therefore, as a conclusion it can be stated that the NJ guide trees used in TC or PTC are very unbalanced.

To sum up, NJ guide trees are unbalanced, because they only take into account the similarities between sequences to generate these trees. The problem of working with unbalanced trees during the progressive alignment stage is that there are too many precedence relations between the tree nodes and this generates longer critical paths. Unbalanced trees also affect the degree of parallelism. The more unbalanced a tree is, the fewer the tasks that can be launched in parallel and thus the lower the degree of parallelism is. In other words, if we have a low degree of parallelism, full performance is not obtained in the HFLET parallel strategy and many computing resources are not used.

3.3 Balanced Guide Tree

This section presents the Balanced Guide Tree (BGT) proposal designed to solve the problems presented in section 3.2. BGT consists of modifying the tree generation method to take into account not only the similarity between sequences, but also balancing features. The aim is to generate better-balanced guide trees than the ones generated with the original NJ heuristic, without losing the alignment accuracy. Finally, the main goal of this balancing heuristic is to reduce the number of precedence relations, decrease the critical path and increase the degree of parallelism.

3.3.1 Neighbor-Joining Algorithm

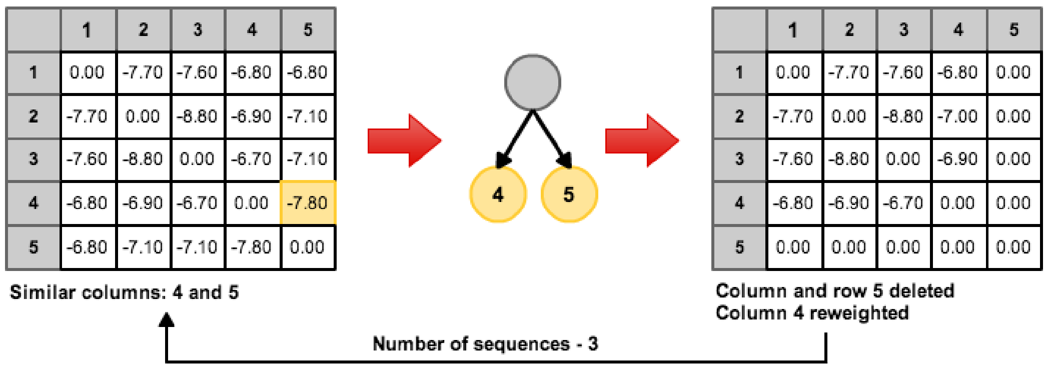
The NJ clustering algorithm consists of three steps:

1. Searching the nearest pair of sequences. The algorithm compares the similarity of each distance matrix column to the other columns and selects the two most similar columns.

2. Grouping this pair of sequences. The two closest sequences are grouped and are linked by the same tree node in the guide tree. Each tree node represents the alignment between the sequences of their child nodes.

3. Replacing pairs by joining similarity. One of the grouped columns is deleted and the other column is filled with the new recalculated similarity values. This recalculated column represents the joining similarity of the group of aligned sequences.

Figure 3.3 shows an example of these three steps. In step 1, the method applies a metric function that determines columns 4 and 5 as the two closest sequences. Then, in step 2, these two sequences are joined with a tree node. Finally, in step 3, row and column 5 are deleted to prevent this sequence being taken into account in the following iterations. The the row and column 4 are filled with a new value that represents the joining similarity of sequences 4 and 5. This method is repeated (#sequences - 3) iterations, and the last three columns are linked directly with the root node.



	1	2	3	4	5
1	0.00	-7.70	-7.60	-6.80	-6.80
2	-7.70	0.00	-8.80	-6.90	-7.10
3	-7.60	-8.80	0.00	-6.70	-7.10
4	-6.80	-6.90	-6.70	0.00	-7.80
5	-6.80	-7.10	-7.10	-7.80	0.00

Similar columns: 4 and 5

	1	2	3	4	5
1	0.00	-7.70	-7.60	-6.80	0.00
2	-7.70	0.00	-8.80	-7.00	0.00
3	-7.60	-8.80	0.00	-6.90	0.00
4	-6.80	-6.90	-6.70	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

Column and row 5 deleted
Column 4 reweighted

Figure 3.3: Example of NJ Guide Tree generation

3.3.2 Balancing Guide Tree Algorithm

The BGT NJ clustering algorithm is derived from the original NJ algorithm to maintain similarity properties and balance the guide tree. BGT tries to join the maximum number of pairs of sequences and locate them at the base of the tree in order to reduce the number of tree levels and thus reduce the critical path. In order to maintain the accuracy of the TC algorithm, the balancing heuristic is only applied if two sequences are quite similar.

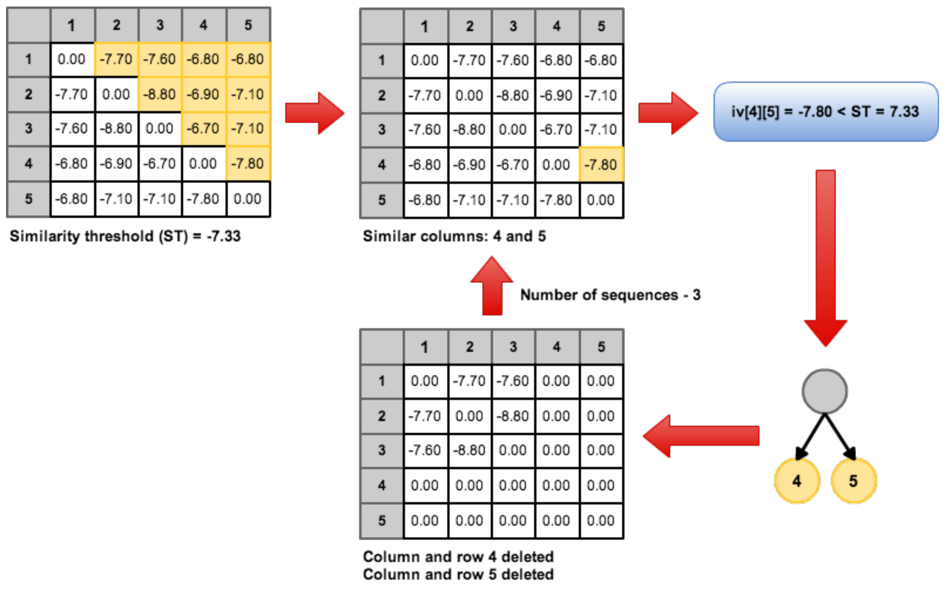
To decide when to apply the BGT or the original heuristic, we use the similarity threshold. This threshold is obtained by calculating the mean distance among all the sequences to be inserted into the guide tree. This average defines a similarity threshold in the sense that the BGT method will not group any pair of sequences whose distance is bigger than this value (i.e. low-related sequences).

To implement the BGT heuristic, the main key changes performed to the original heuristic are:

1. Calculating the similarity threshold. This is the average value between the similarity values of the upper diagonal in the distance matrix (Figure 3.4 step 1). It is used to decide which method to use to group a pair of sequences: the BGT NJ heuristic or the original NJ heuristic.

2. Searching and grouping the nearest pair of sequences. Like the original NJ algorithm, the nearest pair of sequences are grouped. The main difference is that if the intersection value between the pair of sequences in the distance matrix satisfies the constraint imposed by the similarity threshold, this pair of sequences is grouped and their respective columns and rows in the distance matrix are deleted (Figure 3.4 step 2). If not, they are grouped using the original NJ heuristic. This stage is repeated (#sequences - 3) iterations, and the last three columns are grouped directly with the root node.

3. Replacing pair by joining similarity. At each iteration, the joining similarity values of these columns are calculated as the original method. The similarity values will be replaced when one of these conditions is



	1	2	3	4	5
1	0.00	-7.70	-7.60	-6.80	-6.80
2	-7.70	0.00	-8.80	-6.90	-7.10
3	-7.60	-8.80	0.00	-6.70	-7.10
4	-6.80	-6.90	-6.70	0.00	-7.80
5	-6.80	-7.10	-7.10	-7.80	0.00

Similarity threshold (ST) = -7.33

	1	2	3	4	5
1	0.00	-7.70	-7.60	-6.80	-6.80
2	-7.70	0.00	-8.80	-6.90	-7.10
3	-7.60	-8.80	0.00	-6.70	-7.10
4	-6.80	-6.90	-6.70	0.00	-7.80
5	-6.80	-7.10	-7.10	-7.80	0.00

Similar columns: 4 and 5

	1	2	3	4	5
1	0.00	-7.70	-7.60	0.00	0.00
2	-7.70	0.00	-8.80	0.00	0.00
3	-7.60	-8.80	0.00	0.00	0.00
4	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00

Column and row 4 deleted
Column and row 5 deleted

Figure 3.4: BGT Guide Tree generation example

achieved: 1) all the rows and columns have been deleted or, 2) the intersection value between the pair of sequences fails to satisfy the constraint imposed by the similarity threshold.

Three different approaches were defined allowing the users to prioritize which parts of the tree they want to balance, which parts they want to maintain the similarity features in and which they want to use one heuristic or another in. These three approaches are briefly explained below:

- AllBalancing: In this first approach, the BGT heuristic is used to generate the whole tree to obtain a balanced tree taking similarity features into account.

- LeafBalancing: In this second approach, BGT is only used to generate a percentage n of the sequences of the tree, and the rest of the tree is generated with the original NJ heuristic. With this approach, the base of the tree is balanced, taking similarity features into account and the tree nodes near the root are unbalanced.

- RootBalancing: In this last approach, BGT is only used to generate a percentage n of the sequences of the tree, and the rest of the tree is generated using an approach that only considers balancing features. With this approach, the base of the tree is balanced, taking similarity features into account and the tree nodes near the root are perfectly balanced.

Figure 3.5 shows the same tree as Figure 3.2, but in this case, generated with the AllBalancing BGT approach. This tree is better balanced than the previous one. Thus, comparing it with the tree in Figure 3.2, it can be noticed that the critical path is shorter and the degree of parallelism is greater. The critical path length is reduced by 43%, from 7 to 4 iterations, and the degree of parallelism is increased from 3 to 5 tasks, 66% greater.

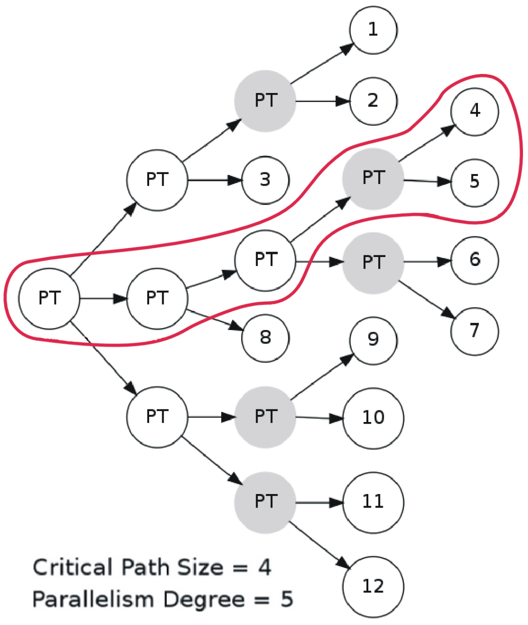


Figure 3.5: Guide tree generated with the BGT-AB heuristic

Tables 3.2a and 3.2b show a similar study to the one in Table 3.1. However, in this balancing study, the critical path and the maximum parallelism degree of the original NJ clustering algorithm are compared to the three proposed approaches: AllBalancing BGT (BGT-AB), LeafBalancing BGT using a percentage n of 25 (BGT-LB25), and RootBalancing BGT with a percentage n of 25 (BGT-RB25).

Sequences	NJ	BGT-AB	BGT-LB25	BGT-RB25
PF00074	28	11	26	9
PF00349	23	19	23	9
PF00231	26	18	26	10
PF01057	94	14	93	9
PF00200	29	17	28	10
PF08443	68	29	62	10

(a) Critical path lengths

Sequences	NJ	BGT-AB	BGT-LB25	BGT-RB25
PF00074	107	172	129	174
PF00349	143	252	183	257
PF00231	164	270	191	277
PF01057	71	187	122	191
PF00200	173	295	218	297
PF08443	215	365	278	374

(b) Maximum degree of parallelism

Table 3.2: The NJ and BGT approaches features for some sequence sets from the Pfam database

The results in Tables 3.2a and 3.2b show that both BGT-AB and BGT-RB25 are able to reduce the critical path length and increase the degree of parallelism considerably, obtained better balanced trees. However, BGT-LB25 obtains similar characteristics to the original one, because only the base of the tree is balanced. Furthermore, BGT-RB25 generates the guide trees with the shortest critical path and the greatest degree of parallelism. On average, this approach reduces the critical path by 71.92% and increases the degree of parallelism by 91.57% compared to the NJ Trees. The extreme case appears with the PF01057 data set, where BGT-RB25 reduces the critical path by 90.42% and increases the degree of parallelism by 196.01%. The reason for these results is that in BGT-RB25, the base of the tree is balanced as in the BGT-AB approach, however, the nodes near the root are perfectly balanced.

The conclusion extracted from these results is that guide trees generated with BGT approaches are better balanced than guide trees obtained with the original NJ clustering algorithm.

Finally, BGT approaches were implemented in PTC, the method being called BalancedParallel-TCoffee (BP-TC), and also in TC, the resulting method being called Balanced-TCoffee (B-TC). However, a standalone version of BGT has also been implemented to take advantage of balanced trees in any method that accepts a NJ tree as an input parameter. The main difference between the integrated implementations and the standalone one is that while integrated versions use the distance matrix from the TC or PTC methods, the standalone version uses any distance matrix produced by any method. Furthermore, the standalone version is also capable of rebalancing a guide tree, which is introduced as an input parameter by the user.

3.4 Experimentation

This section presents the BGT experimental results to verify the improvement in execution time due to balanced guide trees and validate that the proposed approaches maintain the same level of biological accuracy as the original TC. Therefore, this section is divided into the following studies: The first study, which is presented in Section 3.4.1, is based on a performance comparison between the original PTC and TC methods and the BGT heuristic implemented in both PTC (BP-TC) and TC (B-TC). In the second one, in section 3.4.2, the biological quality of the alignments is analyzed.

3.4.1 Balanced Guide Tree Performance

The following experimentation presents the performance study of the proposed methodology and its approaches compared to the standard one. In summary, this study is divided into two parts depending on the method used. The first one compares the execution times of the PTC to BP-TC approaches running on a distributed machine, in order to analyze the performance improvements. The second one is similar to the first one, but it compares the execution times of TC to B-TC running on a multi-core machine.

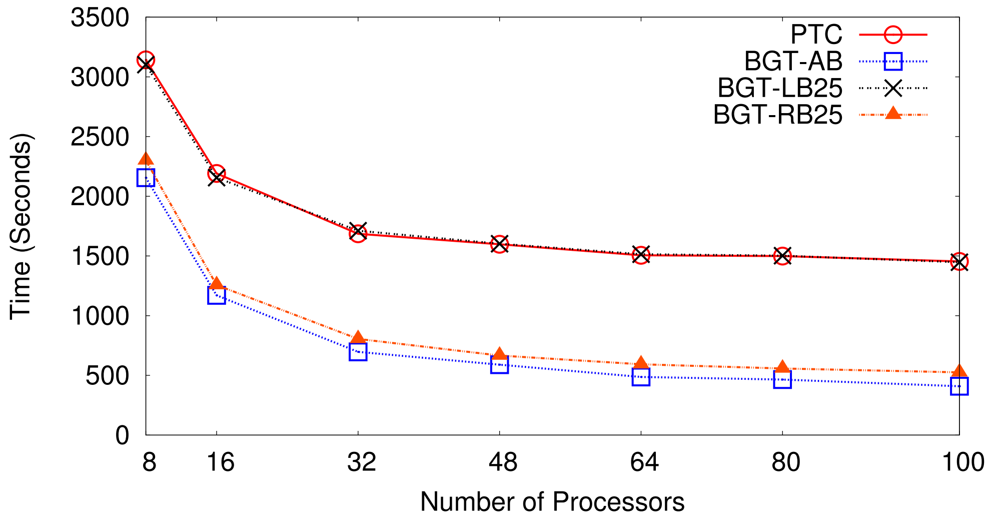
3.4.1.1 BalancedParallel-TCoffee

This first experiment consisted of calculating the execution times of the BGT approaches implemented in the PTC (BP-TC), increasing the number of processors of the cluster. Then, these results were analyzed and compared to the PTC execution times to study the performance and scalability improvements of BP-TCoffee and its approaches, BGT-AB, BGT-LB25 and BGT-RB25. The experiment was run on 8 to 100 processors in a cluster where each node was an HP Proliant DL145-G1, with 2 processors AMD Opteron (1.6GHz/1MB) and 1GB of RAM Memory. The experiment was executed different times using some sequence sets from the Pfam database and the average values were calculated.

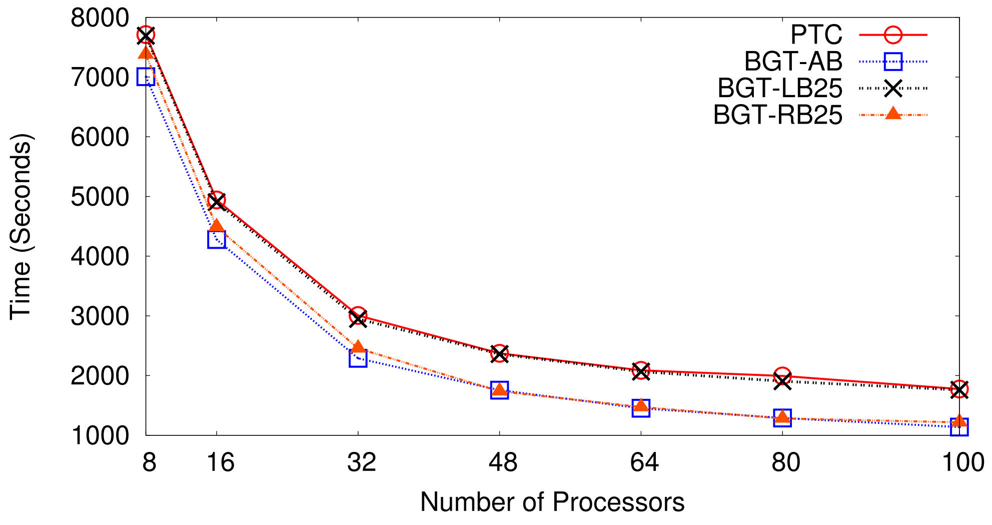
Figure 3.6a shows the comparison of the total execution time using only the PF01057 data set. It can be seen that BGT-AB and BGT-RB25 improve the PTC and BGT-LB25 execution times. With regards to PTC, the biggest improvement was obtained with the higher number of processors used. With 100 processors, BGT-AB reduced the execution time of PTC by 68%. This means that the higher level of degree of parallelism of our approaches can take advantage of large parallel architectures.

It is also remarkable that although BGT-RB25 achieves the shortest critical path and highest parallel degree, it does not obtain the best execution time. This is due to the parallel tasks obtained being more computationally complex which meant the parallelism benefits were reduced. Therefore, BGT-AB is the approach chosen for further performance experiments because the performance is good enough and the balancing is not as aggressive given that it takes accuracy measures into account.

Figure 3.6b shows the average execution time of all the Pfam data sets used in the experiment, which were the same as those used in Table 3.1. Note that similar behaviour as Figure 3.6a was obtained. The total execution time using BGT-AB and BGT-RB25 was lower than the PTC and BGT-LB25 times, because the progressive alignment stage had been reduced. However, it was observed that the percentage of improvement for all sets is up to 36%. This reduction is attributable to the shorter size of some Pfam datasets, which makes BGT less efficient due to less parallelism being required.



(a) PF01057 sequence



(b) Pfam database average

Figure 3.6: Comparison of the PTC and BP-TC execution times

On the other hand, BGT-LB25 did not improve PTC execution time because, as mentioned in previous sections, this approach only balances the nodes near the base of the tree and the nodes near the root are kept in their original form, i.e. unbalanced.

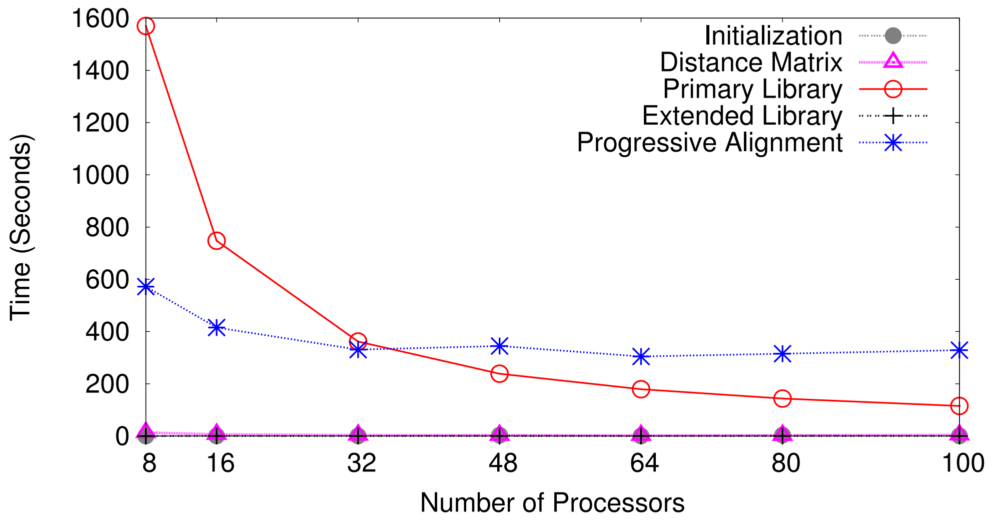
Figures 3.7a and 3.7b show the BGT-AB and PTC execution times for each stage respectively. The results demonstrate that the BTG-AB and BGT-RB25 improvement came from the higher degree of parallelism obtained by the balanced tree. More precisely, BGT reduced the execution time of the progressive alignment stage by more than 64% (from 1,560 to 572 seconds) with 8 processors and more than 75% when the parallel version achieved its best performance (from 1,334 to 329 seconds, in the range of 32-100 processors).

Although Figures 3.7a and 3.7b show that the BGT-AB progressive alignment stage was faster than the same stage in the PTC; from Figure 3.7a, one can also see that the progressive alignment stage remained linear and did not scale like the library stage. That is because the strong dependencies of the tree limited the scalability to the degree of parallelism and also due to the weaknesses in the PTC profile alignment algorithm that are explained in the further sections below.

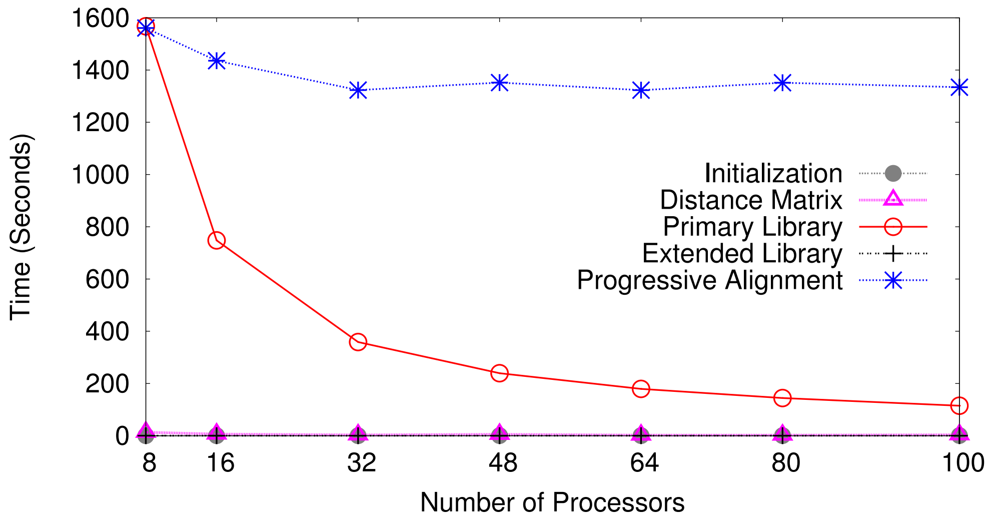
3.4.1.2 Balanced-TCoffee

The second performance study consisted of running the BGT-AB heuristic implemented in concurrent TC (B-TC) on a multi-core machine and varying the number of sequences. The execution times were then calculated and compared to the TC in order to analyze its performance and constraints. This experiment was run on a 24 cores workstation of 32 GB of RAM.

Figures 3.8a and 3.8b show the total execution time (Total Time) and progressive alignment stage time (PA Time) for TC and B-TC BGT-AB. These tests were done using two prefabricated Pfam sequence sets and increasing the number of sequences from 100 to 600 and from 100 to 500 respectively. While Figure 3.8a shows slight performance improvements in the BGT-AB, especially when the number of sequences grows, Figure 3.8b shows that, with this sequence datasets, the BGT-AB approach did not reduce the total execution time because it did not decrease the progressive alignment time of TC.

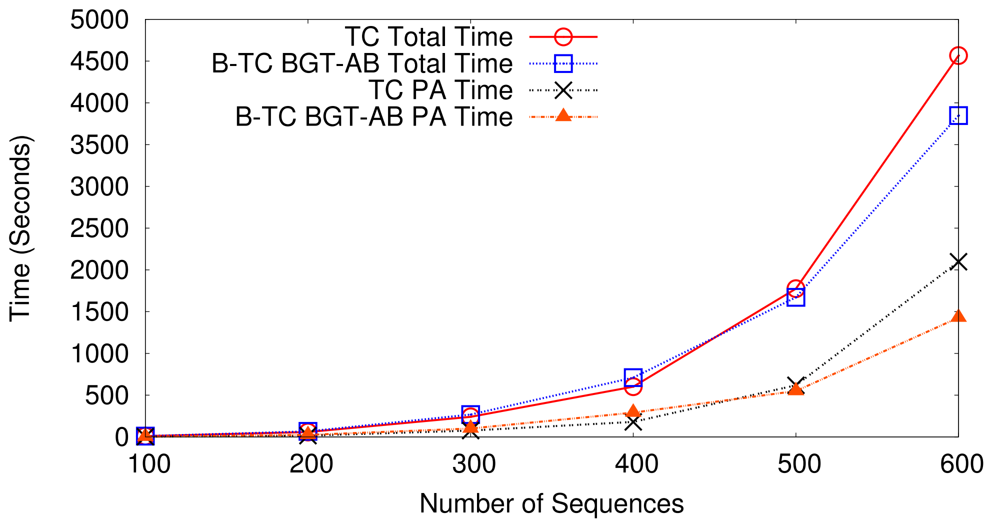


(a) BP-TC BGT-AB stages execution time

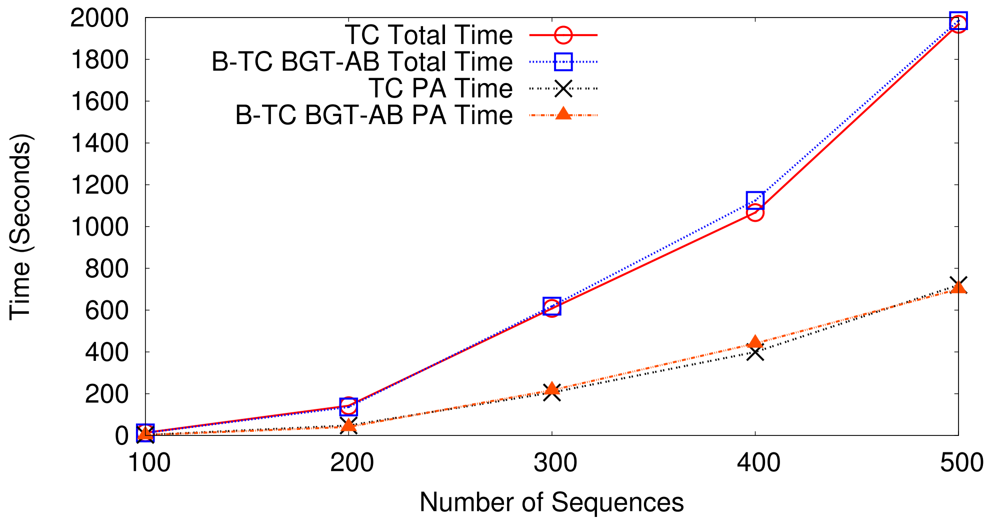


(b) PTC stages execution time

Figure 3.7: Comparison of the stages of the execution times for the PF01057 sequence set



(a) PF0007 sequence set



(b) PF1057 sequence set

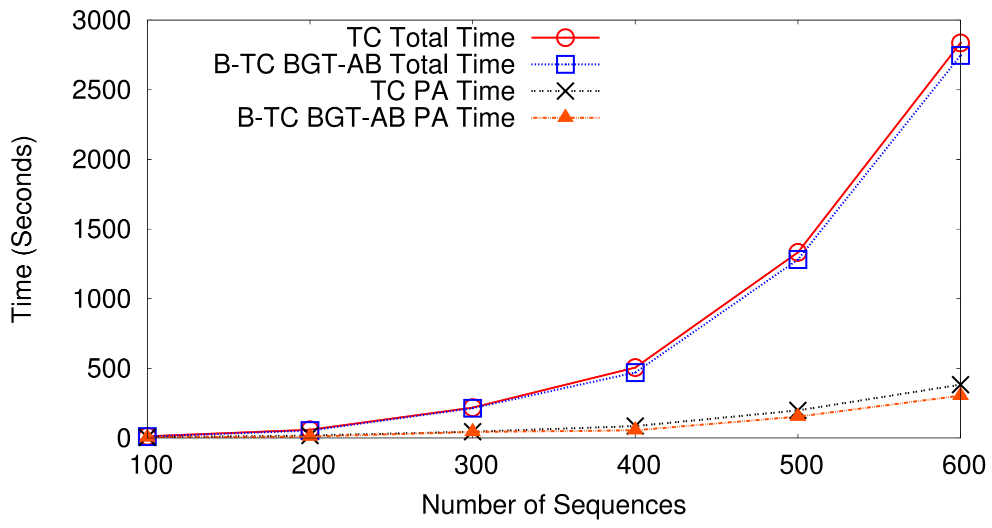
Figure 3.8: Comparison of the PTC and B-TC execution times

These variable results do not show the improvements in the degree of parallelism obtained by BGT because the pairwise profile alignment algorithm used by TC and PTC to align two profiles during the progressive alignment stage is not thought to treat balanced guide trees. Table 3.3 shows three tested examples where the time to align two profiles, depending on the number of sequences (Nseq Prf1 and Nseq Prf2) of each profile and the tree used (NJ or BGT-AB), was compared. It can be seen that for this algorithm is more time expensive for this algorithm to align a profile of 108 sequences to a profile of 299 sequences (407 sequences in total) than to align a profile of 12 sequences to a profile of 368 sequences (380 sequences in total). The first one needs 201.08 seconds, while the second one needs only requires 41.39 seconds to align them. This table demonstrates the argument stated above and thus, in the final steps, sometimes it is sometimes more time expensive to align a balanced tree than an unbalanced one.

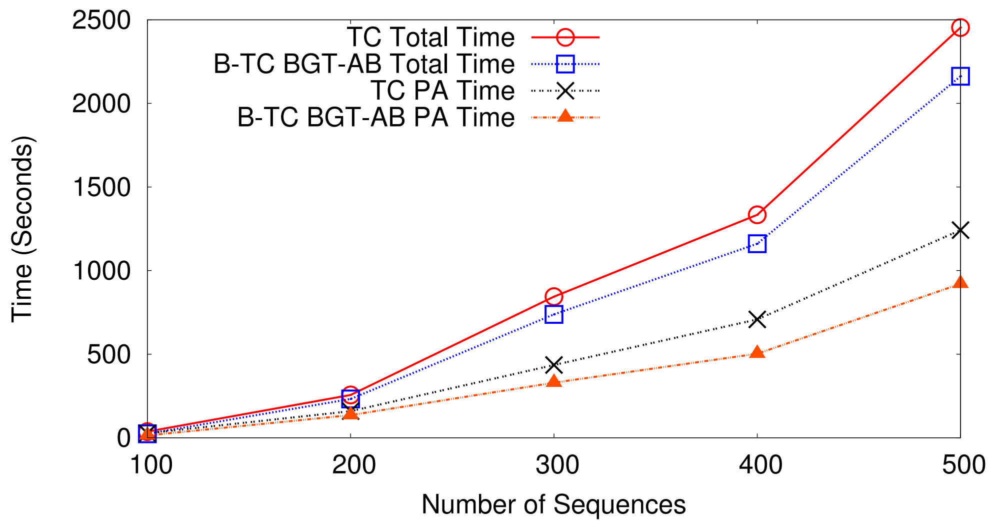
	NJ			**BGT-AB**	
Nseq Prf 1	**Nseq Prf 2**	**Time (s)**	**Nseq Prf 1**	**Nseq Prf 2**	**Time (s)**
1	3	1.61	2	2	4.41
12	368	41.39	108	299	210.08
24	576	106.901	104	496	376.47

Table 3.3: Comparison of the TC and BGT-AB profile alignment times

To attempt to avoid or reduce the above-mentioned problems, Figures 3.9a and 3.9b show the same experiment as Figures 3.8a and 3.8b, but using the Myers and Miller algorithm to align the profiles instead of the default algorithm. The Myers and Miller [MM88] is a linear-space algorithm for producing optimal sequence alignments. Both Figures show that the BGT-AB approach improves the TC execution time due to the improvement in the progressive alignment stage time. For instance, in Figure 3.9b with 500 sequences, B-TC is 12% faster than TC, because BGT-AB improves the progressive alignment stage time by 26%.



(a) PF0007 sequence set



(b) PF1057 sequence set

Figure 3.9: PTC and B-TC execution times compared using the Myers-Miller algorithm

3.4.2 Balanced Guide Tree Accuracy

This experiment evaluates the biological quality of the alignments generated using BGT guide tree approaches. It then compares the accuracy results achieved with the TC ones, in order to check that the proposals presented do not lose too much biological quality and to validate these. The results were not compared to the parallel version PTC, because the latter implemented an old version of TC and the obtained accuracy values obtained are obsolete.

To validate the presented proposal, the multiple alignment benchmarks BAliBASE and PREFAB were used. Both BAliBASE and PREFAB, which are explained in more detail in Section 2.4, are used to identify the strong and weak points of the alignment programs. While BAliBASE compares the reference alignment to the user alignment and calculates the Sum-of-Pairs (SP) and the Total Column score (TCS), PREFAB does the same but using the Q score.

Table 3.4 compares the accuracy comparison of the three BGT approaches with TC using BAliBASE. In BAliBASE, the results are divided into the nine references and a last category for total results. In each of these divisions, the average of both SP and TCS was calculated. The accuracy scores demonstrate that the quality of the alignments generated with BGT approaches was maintained in comparison with TC. In particular, the total TCS row shows that BGT-AB and BGT-LB25 improved the alignment quality of TC by more or less 1%; while BGT-RB25 obtained the same average accuracy as TC. In summary, the differences in accuracy between TC with BGT-AB were in the interval $[-0.002, +0.008]$.

On the other hand, Table 3.5 displays the same comparison as Table 3.4, but using the PREFAB benchmark. The PREFAB results are divided into four ranges depending on the percentage of sequence identity, and a fifth one to show the total average results (0-100). In this case, Table 3.5 indicates that, on average, the BGT proposals maintain or are close to, the alignment accuracy of TC. For instance, BGT-AB improved the total accuracy results of TC by 0.14%, while BGT-LB25 and BGT-R25 worsened these by 0.30% and 0.40% respectively. Regarding B-TC BGT-AB, the accuracy differences with TC were in the interval $[-0.001, +0.006]$.

In conclusion to the results shown in this experiment, the proposal presented did not diminish the accuracy. Finally, we conclude that the best approach is BGT-AB because it provides a good compromise between time and quality.

Reference		TC	BGT-AB	BGT-LB25	BGT-RB25
Ref1	**SP**	0.764	0.764	0.765	0.764
	TCS	0.579	0.581	0.584	0.581
Ref2	**SP**	0.878	0.876	0.878	0.874
	TCS	0.363	0.366	0.370	0.365
Ref3	**SP**	0.785	0.784	0.789	0.782
	TCS	0.392	0.400	0.406	0.380
Ref4	**SP**	0.802	0.806	0.806	0.803
	TCS	0.420	0.430	0.430	0.426
Ref5	**SP**	0.787	0.790	0.788	0.787
	TCS	0.423	0.429	0.421	0.424
Ref6	**SP**	0.806	0.809	0.811	0.807
	TCS	0.419	0.417	0.418	0.424
Ref7	**SP**	0.805	0.811	0.809	0.797
	TCS	0.359	0.365	0.364	0.354
Ref8	**SP**	0.701	0.701	0.700	0.700
	TCS	0.179	0.180	0.181	0.176
Ref9	**SP**	0.741	0.741	0.745	0.741
	TCS	0.484	0.484	0.487	0.486
Total	**SP**	0.785	0.787	0.788	0.784
	TCS	0.402	0.406	0.407	0.402

Table 3.4: BAliBASE accuracy analysis

Reference		TC	BGT-AB	BGT-LB25	BGT-RB25
0-15	**Q**	0.421	0.427	0.417	0.419
15-25	**Q**	0.721	0.721	0.719	0.717
25-35	**Q**	0.876	0.875	0.876	0.866
35-100	**Q**	0.951	0.954	0.950	0.952
0-100	**Q**	0.709	0.710	0.707	0.706

Table 3.5: PREFAB accuracy analysis

3.5 Balanced Guide Tree Conclusions

In summary, we proposed and evaluated a new Balanced Guide Tree heuristic (BGT), based on the neighbor-joining clustering algorithm, for the construction of a guide tree that can achieve a more balanced tree, resulting in a significant reduction in the critical path and an important rise in the number of tasks that can be executed in parallel. This proposal is designed to eliminate the bottleneck generated by the high dependencies between different iterations of the progressive alignment step.

As a result, BalancedParallel-TCoffee can take advantage of large high performance computing infrastructures to reduce the execution time of MSA applications. Furthermore, Balanced-TCoffee is able to improve the performance of T-Coffee and exploit the computing resources of a multi-core workstation.

The experimental results obtained in a cluster of 100 processors showed that BalancedParallel-TCoffee reduced the execution time of previous Parallel-TCoffee by 68%. Moreover, on a 24 multi-core workstation, Balanced-TCoffee improved the execution time of T-Coffee by 12%. Finally, this performance was achieved while maintaining the biological accuracy of the resulting alignment.

Chapter 4

Consistency Library Optimization

4.1 Introduction

The novelty of T-Coffee (TC) is the combination of the consistency-based scoring function COFFEE with the progressive alignment algorithm. TC introduced a library generated using all-against-all pairwise alignments computed with a pair Hidden Markov Models (HMM) in order to reduce the greediness and increase the accuracy compared with most methods based on a progressive strategy. However, the introduction of these improvements increased the CPU and memory consumption exponentially to the number of sequences being processed. In order to reduce the CPU execution time, from version 8.0 on, TC implements multi-core capabilities. Thus, TC is able to obtain the alignment results in less time. However, the library does not diminish because the number of constraints to be stored in the memory is maintained exponential to the number of sequences to be aligned $O(N^2L)$, N being the number of sequences, and L their average length.

This chapter introduces a new library building methodology to optimize and reduce the library size. Library optimization reduces the amount of consistency data stored, allowing a reduction in the execution time for the same number of sequences to be aligned, and an improvement in the scalability of TC by treating a large number of sequences. Furthermore, the proposed methodology tries not to lose all the accuracy improvements gained by the consistency-based scheme and it adapts to the the users' needs.

The remaining sections are organized as follows: Section 4.2 introduces and analyzes the memory and runtime constraints of TC. Section 4.3 presents and describes the proposed methodology. In Section 4.4, experimentation is performed to evaluate the effectiveness of the strategy and finally, the main conclusions are summarized in Section 4.5.

4.2 Problem analysis

TC, which is explained in more detail in Section 2.3.4, is one of the most accurate Multiple Sequence Alignment tool (MSA) that combines the consistency-based scoring function COFFEE with the progressive alignment algorithm. While progressive methods tend to offer a good compromise between quality and time, these methods are far from obtaining an optimal solution due to the propagation of errors made in the early stages of alignment. The solution proposed by TC is to use a consistency-based scheme to correct the misalignments of the progressive alignment heuristics. Consistency-based methods are widely recognized as an important development in the field of MSA, but this improvement comes at a cost. These methods have increased CPU and memory consumption exponentially to the number of sequences being processed.

In TC, the consistency-based scheme is achieved through a collection of pairwise alignments called library. The library is obtained by previous alignments of all sequences using third-party aligners. This is one of the strengths of TC because the library can be produced from several pairwise alignment methods or by combining different MSA tools. Finally, the library is turned into an MSA using a position-specific scoring scheme (COFFEE) derived from the library.

The library, L, can be represented by a NxN matrix (see figure 4.1), where there is a list in position $L(i, j)$ of pairwise residue matches for sequences i and j $(i \neq j)$, called *constraints*. Each entry/constraint is a 3-tuple $\{S_i^x, S_j^y, W_{i,j}\}$, where S_i^x denotes the residue x of sequence i, there is some pairwise alignment or other evidence supporting the alignment of S_i^x with S_j^y, and $W_{i,j}$ is the weight of the constraint. The weight, $W_{x,y}$, used to identify the correctness of a constraint is the *sequence identity*, which is known to be a good indicator

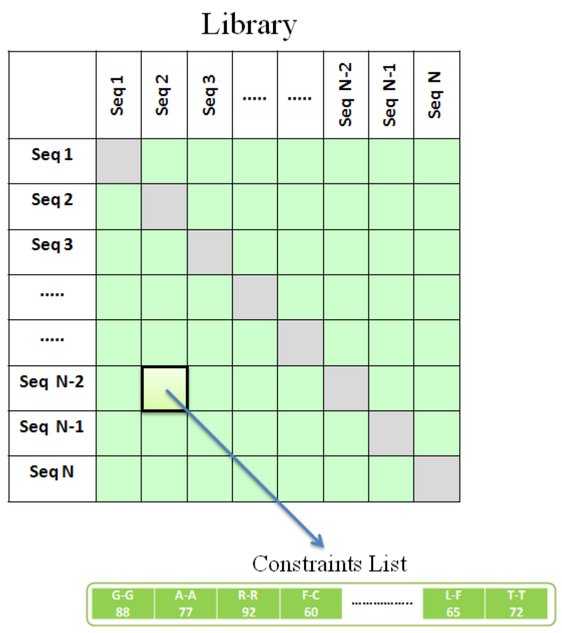


Figure 4.1: Library structure.

of accuracy when aligning sequences with more than 30% identity [SS91]. TC assigns this weight to each constraint in order to give more priority to the most reliable residue pairs. If different pairwise alignment sources are used, then duplicate entries can appear. These duplicated pairs are merged into a single entry that has a weight equal to the sum of the two weights. Pair residues that did not occur have no entry in the constraint list.

The generation of the library requires $(N^2 - 1)$ / 2 pairwise alignments, where N is the number of the sequences. In addition, the progressive alignment stage requires $N - 1$ partial multiple alignments to build the MSA, where each partial alignment can be computing-intensive because the library information is used at this stage. The increasing complexity of the progressive alignment phase, added to the increase in computing due to the generation of all pairwise alignments to build the library, means that the CPU execution time increases significantly, turning T-Coffee into a very slow method compared with other MSA tools and restricting its use to aligning a small number of sequences. For example, T-Coffee is not capable of aligning more than 200 sequences of 500 residues on a standard desktop computer.

Figure 4.2 displays the increase in execution time depending on the number of sequences. It can be seen that runtime requirements also grow quadratically with the number of sequences, demonstrating what is stated above about T-Coffee being a very slow method due to the library construction and progressive alignment stages.

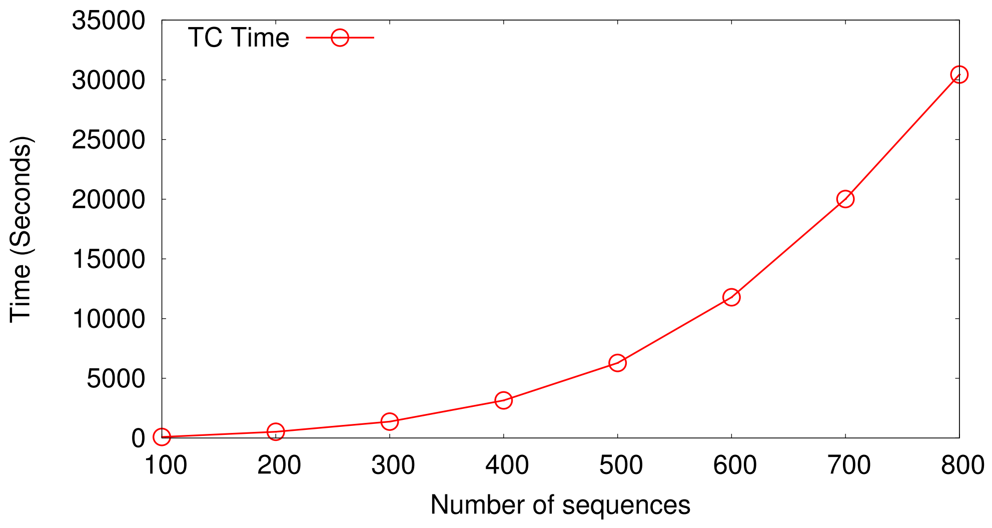


Figure 4.2: Analysis of the TC execution time

On the other hand, the size of the library is proportional to $N^2 * L$, where L is the average length of the input sequences. Figure 4.3 shows the growth of the memory requirements based on the number of sequences and their length. It can be seen that memory requirements grow quadratically with the number of sequences and their length, turning TC into a non-scalable method that is incapable of aligning large numbers of long sequences without saturating the memory.

4.2.1 T-Coffee scalability solutions

In order to reduce the wallclock time requirements, from version 8.0 on, certain stages of TC, like the library generation and progressive alignment, have been parallelized using a multi-process implementation that takes advantage of shared-memory multi-processor/multi-core architectures [TOG+10]. The li-

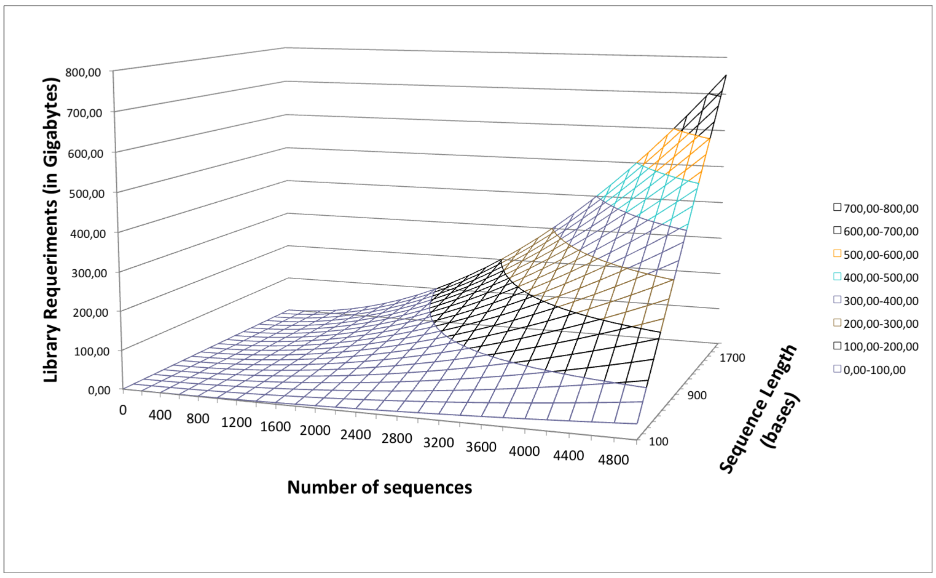


Figure 4.3: Analysis of the TC memory requeriments

brary parallelization is implemented using a master-worker strategy where all the alignment tasks are divided into a number of individual jobs equal to the number of available processors. Jobs are then submitted simultaneously, and their output merged into a single library by the master process. Otherwise, the progressive alignment stage is parallelized by identifying all partial alignment tasks (nodes) in the guide tree that do not depend on other tasks and then processing all these independent nodes in parallel until the final MSA is obtained. This new concurrent version is capable of reducing the execution time, but does not improve the scalability of the method because it does not solve the problem of memory consumption.

TC also incorporates a new library generation method that reduces memory requirements using BLAST [AGM$^+$90] to build it instead of all-against-all pairwise alignments (TC-BLAST). BLAST identifies the most representative sequences for building the library and thus reduces the number of pairwise alignments and the memory requirements. However, as will be seen in the experimentation section (Section 4.4), this approach reduces the quality of the alignments considerably.

In conclusion, TC is not a scalable MSA tool because of its high memory and CPU time requirements, which are introduced by the consistency-based scheme, it being impossible to align large number of sequences. In the next section, a new algorithm to optimize and reduce the high memory demand in order to decrease the CPU execution time and also improve the scalability of TC is presented.

4.3 Library Optimization Proposal

To solve the problems presented in Section 4.2, we present a library-generation optimization method capable of reducing the memory requirements, of the original method in order to decrease the execution time, enhance the scalability of the application, and thus increase the number of sequences that can be aligned.

The methods described in the following subsections are developed in the kernel of TC. As stated in previous sections, TC is a multi-core version where the generation of the library and progressive alignment are divided into tasks to be run in parallel. With the combination of the optimization library method and concurrent execution, the objective is to compute both faster and larger alignments than the standard multi-core TC. Moreover, as this optimization is implemented in the kernel of TC, not only is the default T-Coffee tool able to benefit from these performance improvements, but also all the important tools of T-Coffee package tools, such as M-Coffee, R-Coffee and Expresso.

4.3.1 Library Construction

In current versions, the library is a collection of pairwise alignments obtained from computing all-against-all pairwise alignments or multiple alignments using some predefined method.

The library generation proceeds in two phases: the Primary Library and the Extended Library construction, presented in Algorithms 1 and 2 respectively. The Primary Library provides information about the consistency of each pair of sequences. Meanwhile, the Extended Library improves the consistency, increasing the value of the information in the library by examining the consistency of each pair of residues with residue pairs from all the other alignments.

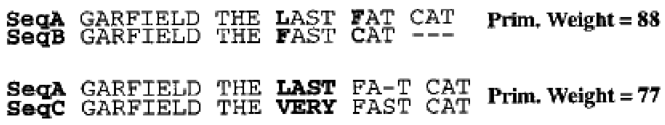
The Primary Library construction, shown in Algorithm 1, evaluates all pairs of sequences S_i and S_j in order to calculate the $W_{(x,y)}$ weight for each matched residue (x, y) (line 5). This weight corresponds to the ratio between the number of matched residues (identical value), calculated by the OCCURRENCE function; and the total aligned residues, calculated by the RESIDUES function.

Algorithm 1 Primary Library construction

```
1. For each sequence $S_i \in S_1..S_N$ and $S_i \neq S_j$
2.    For each sequence $S_j \in S_i..S_N$ where $S_i \neq S_N$
3.       $PA_{ij}$=Pairwise-Alignment$(S_i, S_j)$
4.       For each residue $x \in S_i, y \in S_j|$ are aligned in $PA_{i,j}$
5.          $W_{(x,y)} = \frac{\sum OCCURRENCE(PA_{i,j})}{RESIDUES(PA_{i,j})}$
6.          $L(S_i^x, S_j^y) = L(S_i^x, S_j^y) \cup W_{(x,y)}$
7.       end_for
8.    end_for
9. end_for
```

Figure 4.4 presents an example of the $W_{(x,y)}$ calculation. In the first case, it can be observed that the residues of SeqA and SeqB have the same value, G, in column 1. This means that the residues are *matched.* However, in column 12, the values of the residues compared are different. In some cases, the alignment process inserts gaps, denoted by the symbol '-', to optimize the final result. For this example, the sum of the OCCURRENCE function invocations returns 16 matches and the RESIDUES function returns 18. The gaps are not taken into account in the calculation. For these reason, $W_{(x,y)}$ is 0.88.

The Extended Library construction, shown in Algorithm 2, is based on checking the alignment of all pairs of two residues with residues from the re-



```
SeqA GARFIELD THE LAST FAT CAT    Prim. Weight = 88
SeqB GARFIELD THE FAST CAT ---

SeqA GARFIELD THE LAST FA-T CAT   Prim. Weight = 77
SeqC GARFIELD THE VERY FAST CAT
```

Figure 4.4: Example of pairwise alignment

maining sequences (lines 1 and 2). The main idea is to apply the transitive property to the Primary Library for each triplet of sequences i, j, k. If one residue in sequence i and j is aligned, and the same residue in sequence j is aligned with another in sequence k, then the residues of sequence i and k are also aligned. This way, these residues are re-weighted. For instance, in Figure 4.5, looking at the alignment of sequence SeqA and sequence SeqB through sequence SeqC, it can be seen that SeqA(G) and SeqC(G) are aligned, as are SeqC(G) and SeqA(G). The algorithm concludes that there is an alignment of SeqA(G) with SeqB(G) through sequence SeqC and it associates that alignment with a weight equal to the minimum of $W_1 = W(SeqA(G), SeqC(G))$ and $W_2 = W(SeqC(G), SeqB(G))$. Since $W_1 = 77$ and $W_2 = 100$, the resulting weight is set at 77. In the Extended Library, this new value is added to the previous one to give a total weight of 165 (i.e. $77 + 88$) for the pair SeqA(G), SeqB(G). The complete extension will require an examination of all the remaining triplets.

Algorithm 2 Extended Library construction

```
1. For each sequence $S_i, S_j, S_k \in S_1..S_N$ and $S_i \neq S_j$
2.   For each pair of residues $(x \in S_i, y \in S_j, z \in S_k)$
3.     If $S_{i,j}^{x,y} = S_{j,k}^{y,z}$ then
4.       $L(S_i^x, S_k^z) = L(S_i^x, S_k^z) + \min(L(S_i^x, S_j^y), L(S_j^y, S_k^z))$
5.     end_if
6.   end_for
7. end_for
```

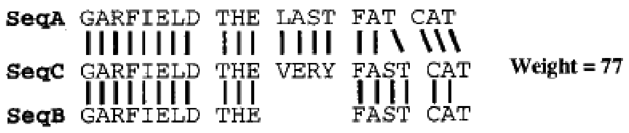

SeqA GARFIELD THE LAST FAT CAT
SeqC GARFIELD THE VERY FAST CAT
SeqB GARFIELD THE FAST CAT
Weight = 77

Figure 4.5: Example of triplet alignment

In summary, the weight associated with a pair of residues will be the sum of all the weights gathered through the examination of all the triplets involving that pair. The more intermediate sequences supporting the alignment of that pair, the higher its weight.

4.3.2 Essential Library Method

The first proposed approach, applied in the Primary Library construction, identifies the information that will be useful during the alignment stage and the information that can be discarded without affecting the quality of the alignment excessively. This consists of building the library in a similar way to the standard method, but identifying those entries in the library that are less representative during the next progressive alignment stage.

The proposed method, shown in Algorithm 3, interprets the sequence identity weight of a constraint (line 5) and compares it with other constraints from the same residue in the library (line 6). If the constraint provides more accurate consistency information, then it replaces the present one in the library.

This phase of the optimization of the library is done during its generation and not after. This way, it builds a smaller library and so TC is able to compute bigger alignments without collapsing the memory, and also reducing the time for building the library and all the steps where the library is used. On the other hand, if the optimization is done when the library has already been built, TC will build a bigger library that may saturate the system before the optimization step. For this reason, this approach is not as scalable as our approach.

Algorithm 3 Essential Library method

```
1. For each sequence $S_i \in S_1..S_N$ and $S_i \neq S_j$
2.    For each sequence $S_j \in S_i..S_N$ where $S_i \neq S_N$
3.        $PA_{ij}$=Pairwise-Alignment$(S_i, S_j)$
4.        For each residue $x \in S_i, y \in S_j|$ are aligned in $PA_{i,j}$
5.            $W_{(x,y)} = \frac{\sum OCCURRENCE(PA_{i,j})}{RESIDUES(PA_{i,j})}$
6.            $L(S_i^x, S_j^y) = \max(L(S_i^x, S_j^y), W_{(x,y)})$
7.        end_for
8.    end_for
9. end_for
```

4.3.3 Threshold Library Method

In the Primary Library, there are a lot of constraints that have little influence on the final alignment. The reason is that due to their weight being so small, they do not provide so much consistency information. In order to identify these constraints, we take the deviation of their weight with regard to the maximum weight in the library into consideration. The deviation is calculated by applying a percentage of this maximum, named *threshold*. The residues with values lower than the one determined by the threshold are discarded.

In Algorithm 4, the threshold verification is applied in line 6. If the $W_{(x,y)}$ is lower than the threshold weight, then it is discarded. However, if it is greater, the same criterion will be applied as in the Essential Library Method. This method allows a further reduction in the library and improves the TC scalability.

It is known that reducing the information stored in the library could affect the final biological quality of the alignment, but the proposed method is able to find a good compromise between the quality and scalability of the application. In addition, our proposed method allows the user to choose how aggressive the reduction of the library size can be. Our implementation allows memory-reduction input parameters to be modified depending on the needs of the moment. The bigger the reduction in the library, the faster it can be built and the bigger, but less accurate the alignments that can be computed. Conversely, the smaller the reduction of the library, the slower it is to build and the smaller but more accurate the alignments can be computed. This relation between performance and accuracy is analyzed in the following section.

Algorithm 4 Threshold Library method

```
1.  For each sequence $S_i \in S_1..S_N$ and $S_i \neq S_j$
2.    For each sequence $S_j \in S_i..S_N$ where $S_i \neq S_N$
3.      $PA_{ij}$=Pairwaise-Alignment$(S_i, S_j)$
4.      For each residue $x \in S_i, y \in S_j|$ are aligned in $PA_{i,j}$
5.        $W_{(x,y)} = \frac{\sum OCCURRENCE(PA_{i,j})}{RESIDUES(PA_{i,j})}$
6.        If $(W_{(x,y)} > threshold \times \max L(*,*))$ then
7.          $L(S_i^x, S_j^y) = \max(L(S_i^x, S_j^y), W_{(x,y)})$
8.        end_if
9.      end_for
10.   end_for
11. end_for
```

4.4 Experimental Study

In this section, we show that our method is capable of enhancing the TC scalability without seriously compromising the biological quality of the alignments. To demonstrate these statements, the following experiments were carried out:

- The first study, which is presented in Section 4.4.1, analyzes the impact of reducing memory in our approach, comparing the results with TC and TC using a library produced by the BLAST method (TC-BLAST).

- The second study is presented in Section 4.4.2 and is based on a performance comparison between TC and one of our approaches.
- In the third and last study, which is presented in Section 4.4.3, the biological quality of the alignments is analyzed. One of our approaches was compared with other MSA tools.

4.4.1 Library Optimization study

The experiment analyzes the impact of the proposed method by varying the input parameters, and it compares the results against the TC and TC-BLAST results. The following input parameters in our method were tested:

- Essential-Library (E-Library): The less representative constraints of each residue are deleted. This method is the least aggressive reduction mode.
- Threshold-Library (T-Library): This configuration tries to remove the less representative constraints from the library, defining a threshold to determine the level of optimization. In this experiment, the levels used are 25, 50 and 75 (T-Library25, T-Library50 and T-Library75), where the smaller this threshold is the lower the reduction in memory will be. In these configurations, the E-Library method is also applied.

This experiment was done by averaging or summing all the results of aligning all the sequences sets provided by the PREFAB benchmark. More specifically, it compares the sum of the size of all the libraries, the length of time to obtain all the alignments and the average quality of each approach.

Figure 4.6 shows the library memory requirements for all the methods analyzed. We can see how the original TC method requires more than 50GBs of aggregate memory for the library for all PREFAB alignments. It is clear that TC-BLAST is the method that requires less memory, 15,000 MB for all the alignments, improving the original TC by an average of 70%.

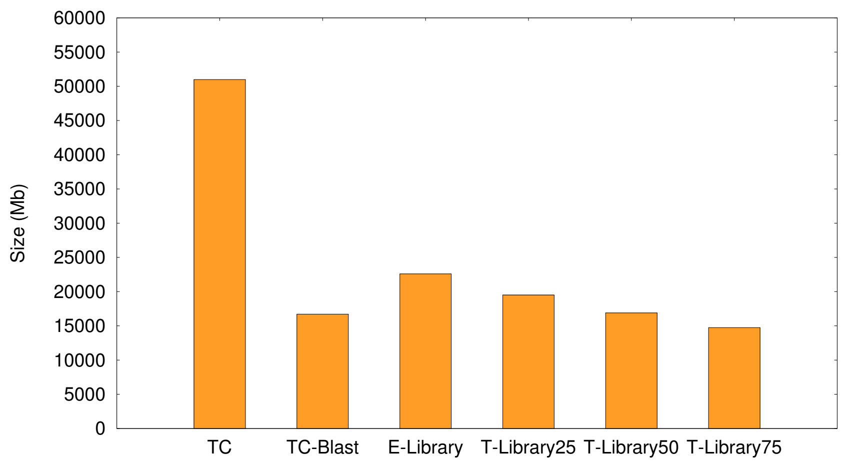


Figure 4.6: Memory requirement analysis

With regards to the optimized methods, we can see that the least drastic one, E-Library, achieves a memory reduction of 61.82% (down to 21 GB). Using both E-Library and T-Library methods together, further optimization can be reached, reducing the needed memory to 18 GB, 16 GB and 14 GB depending on the user level of threshold. It is important to note that T-Library50 is able to obtain the same memory requirements as TC-Blast and T-Library75 improves its requirements by 6.67%. The most drastic configuration only requires 1/4 part of the memory of the original T-Coffee.

Figure 4.7 shows the execution time for the different tests. These results show that our optimization approaches improve the total alignment time of T-Coffee by 27% while TC-BLAST has a more impressive performance, reducing the execution time by 73%. This improvement in performance is because TC-BLAST only needs to perform all-against-all pairwise alignments among a subset of the sequences (the most representative) to calculate the library, while the other methods require all the pairwise alignments to be performed.

Figure 4.8 shows the average accuracy of all these tests. Here we can notice the impact of the library on the quality of the final results. It shows that TC with the completed library obtains the best quality, 0.709 on average. However, the accuracy of the optimized library approaches is not so bad. For all our approaches, the quality ranges from a maximum of 0.699 to a minimum of 0.687. Therefore, in the worst case, the reduction in quality is 3%. The TC-BLAST results are the worst, with a quality of 0.609 and a reduction of 14%.

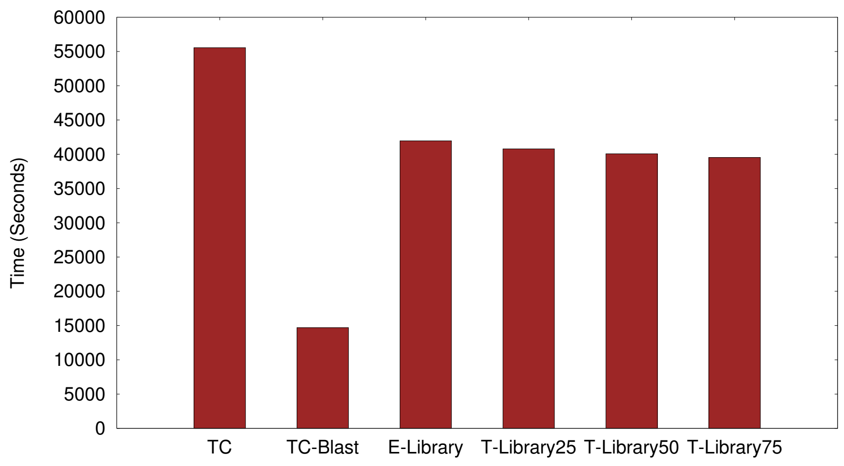


Figure 4.7: Execution time analysis

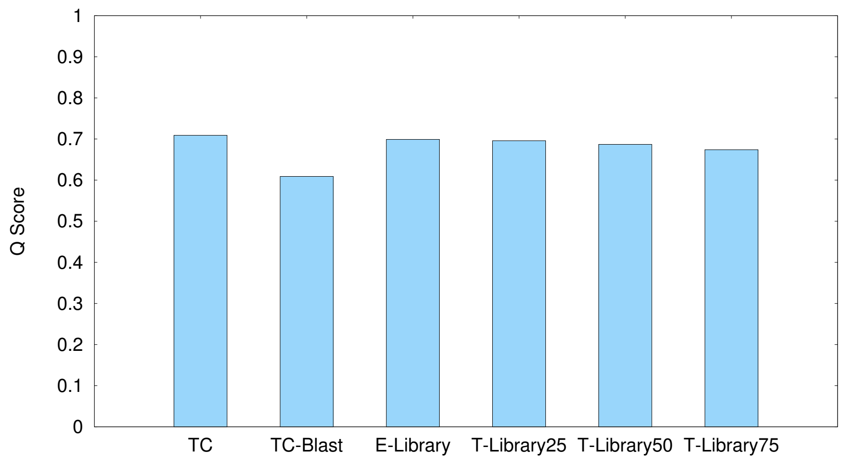


Figure 4.8: Biological accuracy analysis

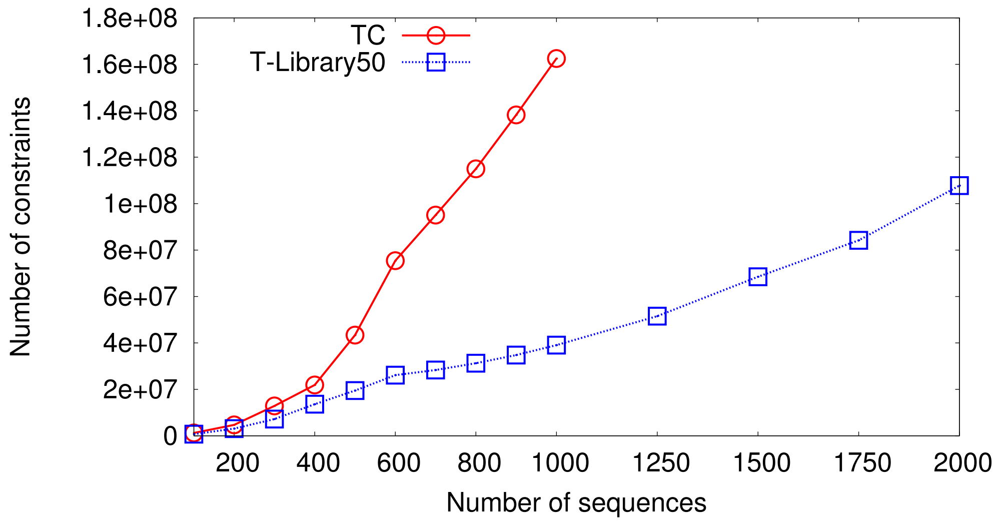
As conclusion, the results presented demonstrate that our approaches (E-Library, T-Library25, T-Library50 and T-Library75) are capable of reducing the size of the library of TC and thus they also reduce the execution time without losing too much alignment accuracy. We believe that the quality lost by TC-BLAST does not compensate for the improvements in time and size.

4.4.2 Scalability study

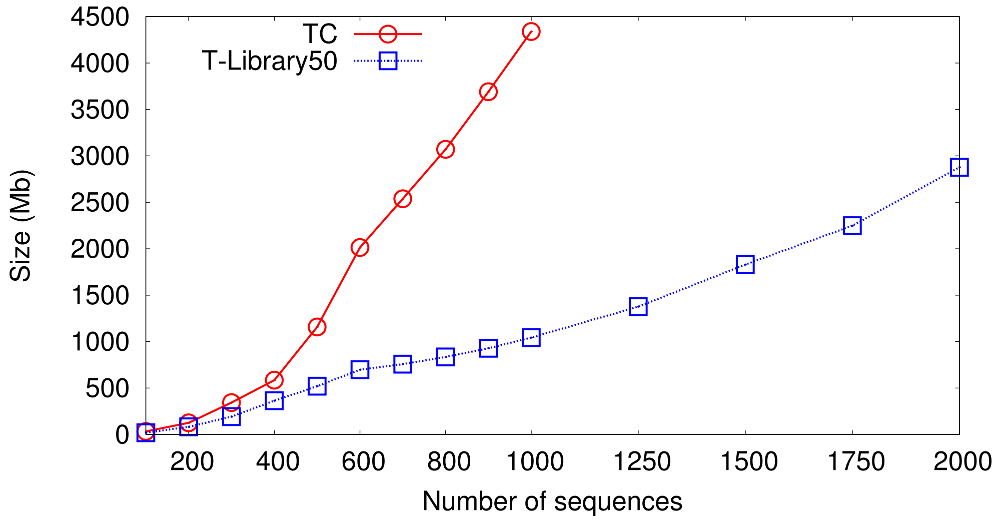
In this experiment, the scalability of our method was analyzed and compared with TC. The experiments were run on HECToR, the United Kingdom National Supercomputer Service and currently the 35^{th} in the top500 list. The HECToR service consists of a Cray XE6 supercomputer, a high-performance parallel file system (esFS), a GPU testbed machine and an archive facility. Each computing node contains two AMD 2.1 GHz 12-core processors giving a total of 44,544 cores, offering a theoretical peak performance of around 373 Tflops. There is presently 32 GB of main memory available per node, which is shared between its twenty-four cores. The total memory is 58 TB. Specifically, a shared memory parallel machine is needed to test the T-Coffee multi-process version. Thus, the experiments consisted of running different sequence sets in a HECToR node, with 24 cores and 32 GB of memory, varying the number of sequences and their length to test the performance improvements of the T-Library50 approach.

Figures 4.9a and 4.9b show the comparison of the number of constraints in the library and its size in Mb respectively, with the number of sequences increasing from 100 to 2000. It can be seen that T-Library50 significantly reduces the number of constraints and the library size compared with TC. With 1000 sequences, T-Library50 reduces the number of constraints and the size of the TC default library by 75%. Due to this reduction in memory requirements, it can be appreciated that with the library optimization approach, TC is capable of aligning up to 2000 sequences while the standard TC is only able to align 1000.

Figures 4.10a and 4.10b show the comparison of the total execution time, the Primary Library (PL) and the Progressive Alignment (PA) times between T-Library50 and TC. It can be seen that our approach is faster than the standard method (Figure 4.10a) because it is able to reduce the PL and PA execution times (Figure 4.10b). With 1000 sequences, TC using T-Library50

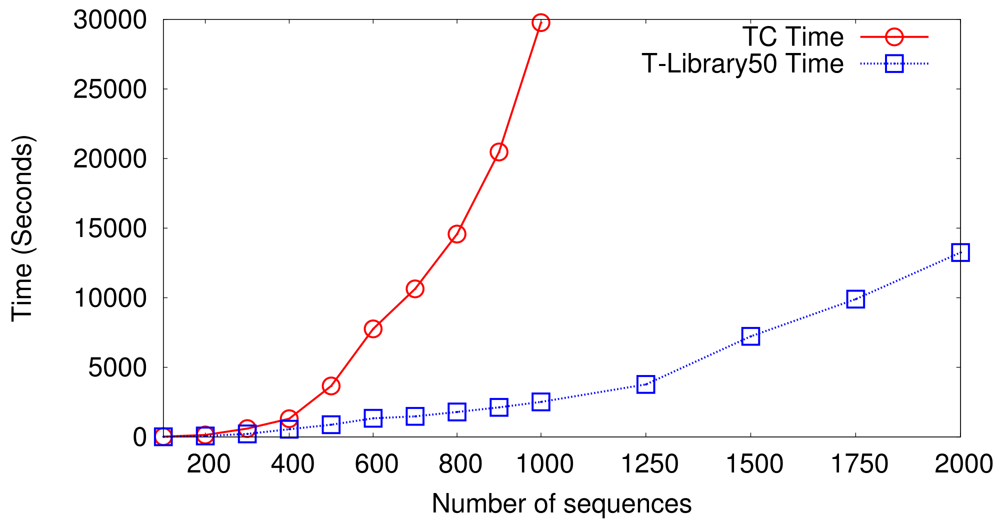


(a) Analysis of the number of constraints

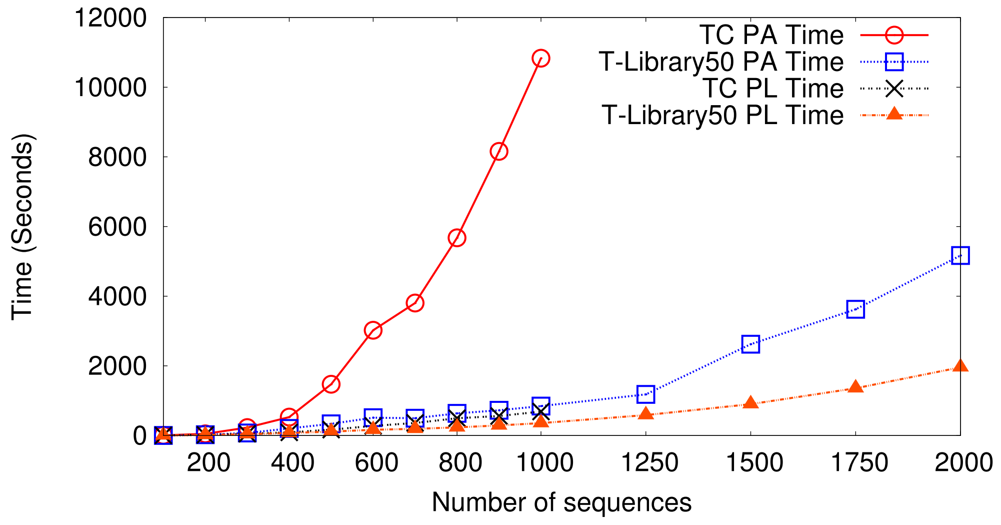


(b) Memory requirements analysis

Figure 4.9: Analysis of the memory requirements



(a) Analysis of the total execution times



(b) Analysis of the stage execution times

Figure 4.10: Scalability study

reduces the execution time of default TC by 92%. We can also see that most of the improvement in the execution times comes from the progressive alignment stage. T-Library50 is able to reduce the execution time of this stage by 92.20%, reducing it from more than 10,832 seconds to only 845 seconds. Meanwhile, the improvement in the library building is not so spectacular as both methods need to calculate all-against-all pairwise alignments. The improvement shown in the graph is mainly due to the data management, which is quicker when the library is more compact.

According to these results, the proposed optimization method reduces the memory and CPU time requirements, thus improving the scalability of TC. Therefore, the resulting method is able to align large numbers of longer sequences.

4.4.3 Biological Accuracy study

This experiment evaluates the quality of the alignments generated with the T-Library50 approach. The results are compared with other related MSA tools, such as the standard T-Coffee (TC), T-Coffee-BLAST (TC-BLAST), ClustalW, Muscle and ClustalΩ. The prepared alignments and the evaluation scores of BAliBASE, PREFAB and HOMSTRAD benchmarks were used to compare the biological quality between the alignments. BAliBASE calculates the Sum-of-Pairs score (SP) and the Total Column score (TCS), and the results are divided into the nine references and a further category for the total average results. PREFAB uses the Q score to evaluate the alignments and the results are divided into four ranges depending on sequence identity percentage, and a fifth one to show the total average results. Finally, the accuracy of HOMSTRAD alignments is calculated with the Column Score (CS) using the *aln_ compare* program from T-Coffee package.

Tables 4.1, 4.2 and 4.3 show the average accuracy results of the methods described above using the BAliBASE, PREFAB and HOMSTRAD benchmarks respectively. It can be seen that the behaviour of our method is similar in the three benchmarks. T-Library50 loses little accuracy compared to the standard method (TC) due to the reduction in the library, but still remains above all the remaining MSA methods, except the new ClustalΩ method, taking into account that this method is only capable of aligning protein sequences. For instance, from PREFAB total average results, it can be inferred that the T-Library50 alignments are respectively 14% and 13% less accurate than those in TC and ClustalΩ, but they are 11% and 1.5% more accurate than ClustalW and Muscle. Moreover, we can also observe that the quality of the results of the T-Library50 were significantly better than those of TC-BLAST. For BaliBASE, the total average SP score improved by 8% and TCS score by 15%. With PREFAB, total Q is improved by 13%, and for HOMSTRAD, the total CS is improved by 9%.

In summary, these results demonstrate that the approach presented represents a better alternative to TC-BLAST for aligning a large number of sequences with the T-Coffee package.

Reference		**T-Coffee**	**TC-BLAST**	**T-Library50**	**ClustalW**	**Muscle**	**ClustalΩ**
Ref1	**SP**	0.763	0.630	0.721	0.667	0.714	0.691
	TCS	0.577	0.429	0.530	0.455	0.519	0.492
Ref2	**SP**	0.876	0.841	0.864	0.822	0.853	0.859
	TC	0.365	0.240	0.334	0.260	0.303	0.336
Ref3	**SP**	0.783	0.686	0.766	0.684	0.750	0.767
	TCS	0.382	0.238	0.361	0.248	0.309	0.386
Ref4	**SP**	0.773	0.739	0.755	0.695	0.759	0.799
	TCS	0.409	0.341	0.385	0.302	0.338	0.428
Ref5	**SP**	0.773	0.739	0.755	0.695	0.759	0.747
	TCS	0.417	0.327	0.383	0.266	0.325	0.355
Ref6	**SP**	0.787	0.724	0.759	0.670	0.732	0.868
	TCS	0.383	0.259	0.333	0.307	0.315	0.572
Ref7	**SP**	0.801	0.683	0.772	0.845	0.787	0.909
	TCS	0.355	0.224	0.318	0.516	0.316	0.624
Ref8	**SP**	0.698	0.673	0.690	0.710	0.685	0.840
	TCS	0.177	0.130	0.168	0.155	0.143	0.437
Ref9	**SP**	0.741	0.696	0.726	0.701	0.715	0.726
	TCS	0.487	0.457	0.464	0.449	0.453	0.476
Total	**SP**	0.778	0.693	0.752	0.717	0.745	0.767
	TCS	0.455	0.356	0.421	0.367	0.399	0.461

Table 4.1: Analysis of BAliBASE accuracy

4.5 Library Optimization Conclusions

T-Coffee is one of the best tools for multiple sequence alignment due to the use of consistency to reduce error propagation in the different steps of the

Reference	**T-Coffee**	**TC-BLAST**	**T-Library50**	**ClustalW**	**Muscle**	**ClustalΩ**
0 - 15	0,421	0,228	0,379	0,289	0,365	0.395
15 - 25	0,721	0,604	0,695	0,605	0,684	0.708
25 - 35	0,876	0,847	0,865	0,816	0,860	0.878
35 - 100	0,951	0,961	0,956	0,941	0,951	0.965
0 - 100	0,709	0,609	0,687	0,617	0,677	0.700

Table 4.2: Analysis of PREFAB accuracy

HOMSTRAD	**T-Coffee**	**TC-BLAST**	**T-Library50**	**ClustalW**	**Muscle**	**ClustalΩ**
Total CS	76.317	68.865	74.992	74.243	75.169	75.766

Table 4.3: Analysis of HOMSTRAD accuracy

progressive alignment. However, the memory and CPU time requirements to implement consistency sometimes discourages users from using this tool to align large numbers of sequences.

In summary, a library optimization method is proposed that improves the scalability and performance of T-Coffee considerably. The proposed method provides the user with greater flexibility to choose how aggressive the reduction of the library can be in order to trade off between alignment time and quality. The library optimization was developed in the kernel of the T-Coffee alignment tool. This mean that not only is the default T-Coffee tool able to benefit from these improvements but also the performance and scalability of all the package tools, like M-Coffe, R-Coffee and Expresso, can be improved.

Furthermore, experimental analysis showed that the method allows larger alignments to be performed than those that can be managed by the original T-Coffee. The improvement presented can widen the range of scenarios in which T-Coffee can be efficiently used as an alignment tool.

Chapter 5

Multiple Tree Alignment

5.1 Introduction

In progressive alignments methods, as explained in Section 2.3.2, the order in which the alignments are performed is determined by a heuristically-build guide tree. Although this heuristic provides a great advantage of speed and simplicity, progressive methods are very dependent on the initial alignments, and several studies have shown that the alignment may be sensitive to errors in the guide tree [PPL$^+$10].

This chapter presents Multiples Tree Alignment (MTA). MTA is a new Multiple Sequence Alignment (MSA) method that consists of creating, aligning and evaluating multiple guide trees in parallel. The main objective of MTA is to cope with the errors from the progressive alignment due to guide tree, in order to improve the biological accuracy. Therefore, MTA is similar to non-stochastic iterative methods as it tries to correct an alignment, but instead of repeatedly re-aligning an initial alignment, MTA aligns multiple modified guide trees simultaneously.

Owing to the fact that MTA produces multiple alignments from multiple guide trees, one of the biggest challenges of this new method is to choose the best alignment. We carried out an extensive comparison study to analyze different evaluation metrics in order to decide which one provides a better correlation between the computational score and biological quality. As a result of this study, this chapter also presents two meta-scores built from a combination of existing metrics by using a genetic algorithm, with the aim of enhancing the correlation between these metrics and thus improving the results of the MTA.

The remaining sections are organized as follows: Section 5.2 introduces the MTA algorithm and its features. Section 5.3 studies the feasibility of MTA and presents the most common scoring functions. Section 5.4 describes the genetic algorithm and the training process for obtaining the two proposed meta-scores. In Section 5.5, experimentation is performed to compare the presented scoring metrics and validate the improvements in accuracy and scalability of the proposed method. Finally, Section 5.7 outlines the main conclusions.

5.2 Multiples Tree Alignment Algorithm

MTA is a MSA method that is able to align multiple guide trees by using an existing MSA program, evaluate them and finally select the one that provides the most accurate results. The MTA method can be applied to any progressive aligner that accepts guide trees as an input parameter. For these aligners, MTA is capable of improving their accuracy without modifying the original methods. In addition, the method also allows different evaluation metrics for selecting the best guide tree. The aim of this approach is to find a variation of the original tree that provides a more accurate alignment than the original. Figure 5.1 shows the algorithm scheme of our proposal. It is made up of four main steps: the Distance Matrix building, the neighbor-joining guide tree generation, the alignment stage and the evaluation phase.

1. **Distance Matrix construction step.** This first step consists of constructing the Distance Matrix (DM), a K×K two-dimensional array, K being the number of sequences. To build it, first, all the distances between all the input sequences have to be calculated. Although there are different methods for calculating distances, the most widely-used one is to perform all pairwise alignments between all sequences. However, KTUP distance heuristic is used because, despite being less sensitive, it is faster than the others. Then, these distances are organized forming the DM, which is needed during the next step to build the neighbor-joining guide trees (NJ).

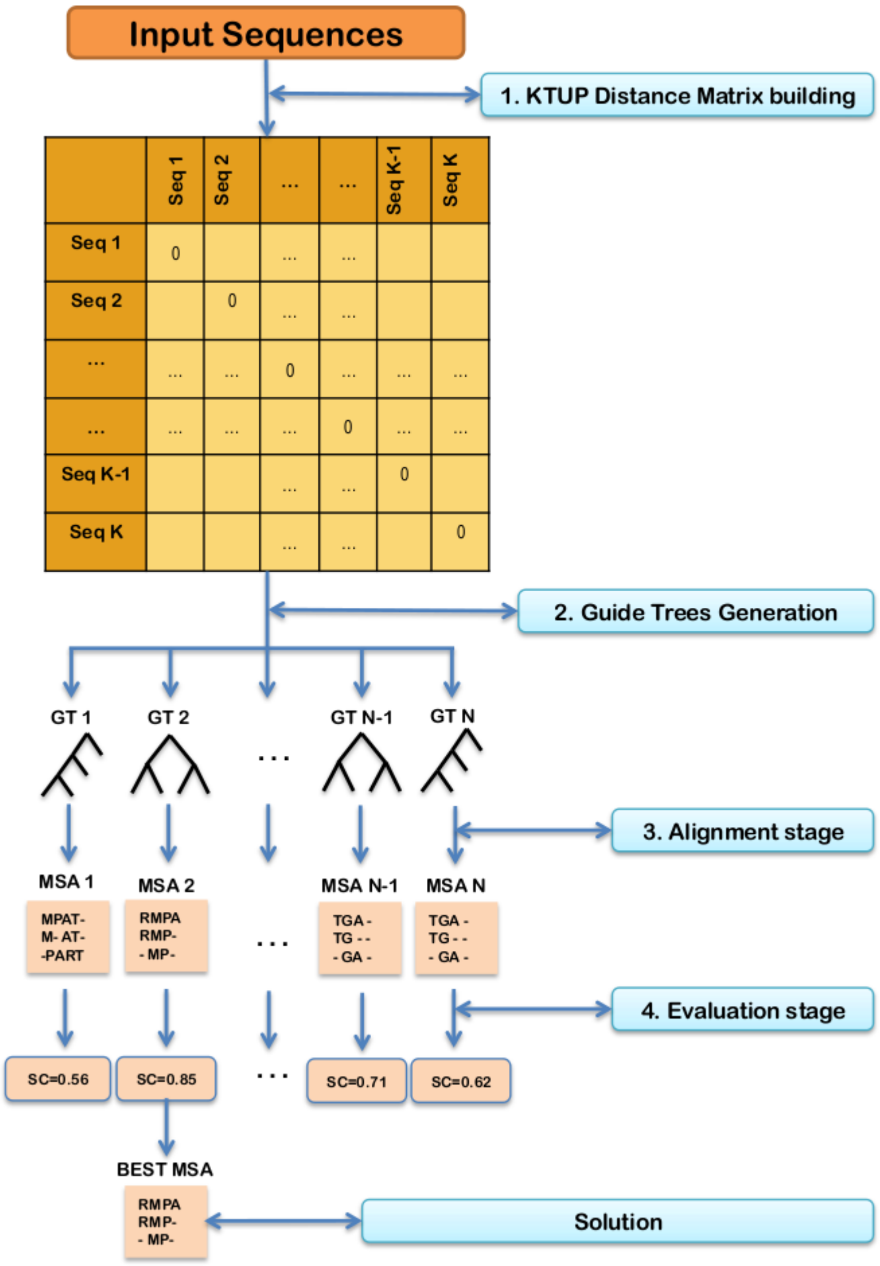


Figure 5.1: Multiple Tree Alignment algorithm scheme

2. **Guide tree generation step.** During this step, the method produces N guide trees, N being defined by the user. The algorithm generates multiple guide trees based in the NJ clustering algorithm. Each tree corresponds to a variation of the original obtained by NJ but adding some noise in the distances in order to introduce some variability. This random noise is low enough to maintain the distance criteria but significant enough to provide the necessary flexibility to generate multiple alternative trees.

3. **Alignment step**. The alignment stage consists of aligning the different guide trees. The method calls an external MSA application that accepts a predefined NJ tree as input parameter. Currently, the alignments are built using two popular progressive alignment applications, namely T-Coffee (MTA-TC) and ClustalW (MTA-CLW). The user is able to select one of them taking the execution time or alignment accuracy criteria into account. T-Coffee (TC) is slower and consumes more memory than ClustalW (CLW), however, it is more accurate as it introduces consistency information to the alignments. MTA is able to share information between any aligner invocations, thus when the selected aligner is TC, the consistency library is calculated once and shared by all the aligner invocations. This optimization allows the execution time required for aligning N guide trees to be reduced and also requires less memory.

4. **Evaluation step.** During the evaluation step, the obtained alignments are scored using an evaluation function/metric in order to identify the best alignment that corresponds to the best guide tree. The alignment with the best score is the final result of the MTA method. As in the alignment step, the method calls an external evaluation score. This allows the user to choose between different evaluation functions. A meta-score, composed of a combination of metrics and generated by a genetic algorithm, can also be used. This part of the algorithm is crucial to achieving improvements in the alignment accuracy, but it is still a challenge to solve for the biologist. Section 5.3 describes the main characteristics and constraints of this step, and then introduces the evaluation metrics

implemented in the literature. Finally, Section 5.4 presents two proposed meta-scores obtained from a combination of existing metrics through a genetic algorithm training.

In contrast with iterative methods, that evaluate the different trees sequentially (one-after-another), the proposed method could evaluate all trees in parallel, thus limiting the time consumed by the method. For this reason, in these early versions, MTA is also implemented to provide an option to run in parallel on a cluster of computers. The guide tree generation, alignment and evaluation steps are parallelized with MPI using the Master-Worker paradigm so they can be implemented as independent tasks. Thus, the master sends the input sequences and the DM to the workers. Each worker then builds a guide tree, aligns this guide tree, evaluates the alignment and send the resulting score to the master. Finally, the master selects the best alignment based on the alignment scores received.

5.3 Evaluation Metrics Analysis

Since the success of the sequence analysis is highly dependent on the reliability of the alignments, measures to assess the quality of the alignments are a high requisite. Thus, in the literature, multiple scoring functions are being developed. Any scoring function needs to take into account that some positions are better conserved than others, which is called position-specific scoring, and that the sequences are biologically related through a phylogenetic tree.

However, as explained in Section 2.4, MSA programs are validated and compared using reference sequence alignment databases. The alignment accuracy is assessed by comparing the test alignment with a reference alignment and calculating a scoring function that measures the proportions of correctly aligned residue pairs or alignment positions in the alignment.

Although some scoring functions introduce sequence structure or function information to evaluate alignments, a perfect function can only determine the mathematically optimal alignment, but not always the biologically optimal one. Therefore, the lack of proper alignment validation methods that could be used without reference alignments is a disadvantage in benchmarking the MSA programs or in determining the best one from a set of alignments.

So, a crucial challenge for the proposed method is to identify the best alignment among those obtained by MTA. However, it is possible that the best alignment chosen through an evaluation metric is not the same as the best biological alignment.

In this section, a study is done first to validate whether the designed methodology in MTA is able to enhance the alignment accuracy. Then, the most common scoring functions are introduced to indicate finally which ones should be implemented in MTA.

5.3.1 Multiple Tree Alignment validation

To reveal the potential of MTA method to increase accuracy, an experiment was conducted in which it was always able to select the best guide tree by using a benchmark score. The experiment showed the best accuracy that MTA could achieve in function of the number of trees evaluated. It is known that benchmarking scores cannot be used in MTA alignments because they are based on reference alignments, which are not always available. However, the use of this benchmarking scores is useful for validating the MTA method, because they allow to determine the maximum degree to which the quality of the alignments can be improved, depending on the number of trees evaluated.

The experiment (Figure 5.2) shows the alignment accuracy evolution depending on the number of trees treated for both MTA-TC and MTA-CLW. This experiment tries to quantify the error introduced by the guide tree. Both methods used the PREFAB Q Score as the selection metric to determine the best alignment. The experiment consisted of evaluating up to 300 trees and using the whole PREFAB datasets as the input sequences and Q score as the validation score to measure the improvements in accuracy. Moreover, when only a guide tree is used, the first graph value corresponds to the accuracy obtained by the default T-Coffee and CustalW respectively.

Figure 5.2 indicates that, in both MTA configurations, the average alignment accuracy rises as the number of trees analyzed increases. Specifically, it is

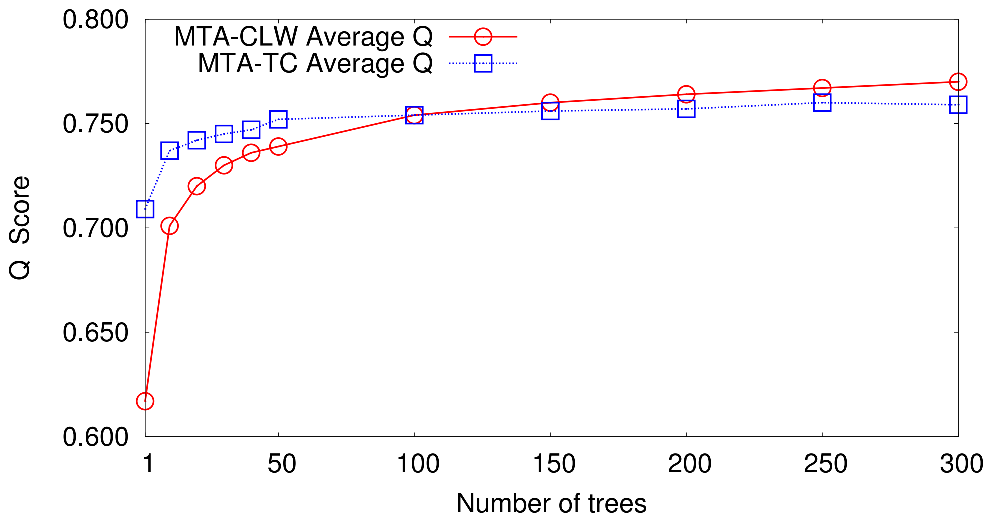


Figure 5.2: MTA validation using Q benchmark score as evaluation metric

shown that, evaluating 300 trees, MTA-TC is able to improve the Q accuracy from 0.709 to 0.759. Furthermore, MTA-CLW improves the CLW alignment accuracy from a Q of 0.617 to 0.779. However, if 5.2 is analyzed in more detail, it can be seen that the largest increase in the accuracy of alignments is achieved with 10 trees. Specifically, in MTA-CLW the accuracy grows from 0.617 to 0.701 and in MTA-TC from 0.709 to 0.737. Then, up to 50 trees, the quality grows progressively, and from this point, the improvement in quality is smaller, at the same time that the number of trees analyzed increases.

Analysing these results, we can conclude that the error introduced by the guide tree is worse in non-consistent methods like ClustalW than in consistency-based methods like T-Coffee. Therefore, consistency is able to correct guide tree building errors. Non-consistency-based methods gap can be estimated at around 13% of the final alignment quality. Another interesting remark is that without taking the error generated by the guide tree into consideration (for example, using MTA with 300 trees), ClustalW, without consistency, is able to produce a better alignment than T-Coffee.

Finally, these results demonstrate that the MTA method is able to improve the quality of an alignment obtained from an MSA method by redefining the guide tree. So now, the major challenge is to find a suitable evaluation metric, without using references, that is able to identify the best tree. Next, the most common evaluation functions implemented in MTA are briefly described to then present the proposed meta-scores.

5.3.2 Scoring Functions

It was demonstrated in Section 5.3.1 that MTA is capable of improving the accuracy of the alignments using an appropriate evaluation metric for discriminating alignments. In the previous test, MTA used the Q score, which needs a reference alignment. Therefore, this cannot be used and a scoring function without reference alignments is necessary to evaluate MTA alignments.

A scoring function is an objective function that assigns a real value to each alignment which should reflect the quality of the alignment. Furthermore, many MSA heuristics work by maximizing an implemented scoring function in order to try to find the optimal alignment. However, as stated in previous chapters, a perfect scoring metric can determine the mathematically optimal alignment, although this is not always the biologically optimal one. Therefore, in our approach, a metric must be found that provides the best correlation between the computational score and the biological quality of the alignment.

Before introducing the most common scoring methods, some of which are implemented in MTA, two different scoring schemes need to be introduced. There are two ways to quantify the sequence similarity: a similarity measure and a distance measure.

- **Similarity scoring scheme:** A similarity scoring scheme is a function that measures the similarity of sequences. In it, the residue-residue pairs are scored with a similarity substitution matrix and residue-space pairs are scored with gap penalties.

  An identity matrix is adequate for nucleic acid alignment. Identity matrices give fixed scores for matches and mismatches. Otherwise, for protein sequences, substitution matrices are used (BLOSUM, PAM, etc). These similarity-based matrices do not contain evolutionary information.

  The gap penalty function is very important since it has a great influence on the distribution of spaces in the alignment, which subsequently affects the overall alignment. In biology, insertion or deletions cause big gaps in an alignment, therefore the most widely used gap penalty model encourages the extension of gaps rather than the introduction of new gaps. This model defines a gap opening penalty and a gap extension penalty, where the gap opening value is always bigger than the gap extension value.

- **Distance scoring scheme:** A distance-scoring scheme, has become more and more important. It is a function that measures the difference between sequences. A key challenge in this approach is how to define a distance function that satisfies the metric property and reflects the evolutionary information between sequences as well. Instead of using a similarity substitution matrix, this scheme uses a distance function to include evolutionary information. For example, Metric PAM250 (mPAM250) consists of a reworking the mathematics of PAM 250 to make its values reflect evolution information from a distance perspective [XM04].

Furthermore, there are many different scoring functions based on using various properties of the sequences to evaluate them. Scoring functions can incorporate the sequences structure, function and evolutionary history to be more accurate.

In the next section, the main scoring functions are briefly described divided into two categories: scoring functions without using structural information and scoring functions using structural information.

5.3.2.1 Scoring Functions without Structural Information

This category introduces some of the most common scoring functions that not use structural sequence information to evaluate alignments.

- **Sum-of-Pairs score (SP):** SP is a pairwise base function introduced by Pearson in 1988 [PL88] and it is the standard method for scoring an alignment. A score is calculated for each pair of sequences in the multiple alignments based on the similarity between the sequences. The score for the multiple alignments is then taken to be the sum of all the pairwise scores. The score of each pairs of sequences is determined by a similarity substitution matrix and gap penalties.

Although the SP score is easy to work with, it is widely used and the results are reasonably good, there is no probabilistic justification for it and each sequence is treated as if it were directly related to all other N-1 sequences, where in fact it is very probably only directly related to one of them. This problem arises because SP does not take advantage of phylogenetic trees.

To compensate for these disadvantages, Carrillo and Lipman proposed a weighted SP score [CL88]. These weights change the importance given to different pairs of sequences and try to reflect known evolutionary distances.

- **NorMD score:** Introduced by Thompson in 2001 [TPR$^+$01], NorMD is a column-based objective scoring function for MSA, based on the Mean Distance (MD) score, that combines the advantages of the column scoring techniques with the sensitivity of methods incorporating residue similarity scores using substitution matrices. MD scores are based on the mean pairwise distance between sequences in a continuous sequence space. NorMD normalizes it to take into account the set of sequences to be aligned.

  The NorMD process consists of calculating a score for each column in the alignment using the concept of continuous sequence space introduced by Vingron and Sibbald [VS93]. The column scores are then summed over the full length of the alignment. In addition, NorMD incorporates sequence information, such as the number and length of the sequences in the alignment set, and the potential sequence similarity.

- **COFFEE score:** The COFFEE score (Consistency based Objective Function For alignmEnt Evaluation) proposed by Notredame in 1998 [NHH98] is a pairwise base function that calculates the correlation between a multiple sequence alignment and a previously defined library of pairwise alignments.

The COFFEE score requires two components: a set of pairwise reference alignments, such as a TC library, and the objective function that evaluates the consistency between an MSA and the pairwise alignments contained in the library.

The COFFEE function presents some similarities with the Weighted SP. As well as SP, COFFEE considers all the pairwise substitutions in the multiple alignments, and weighs these in a way that reflects the relationships between the sequences. However, in this case, the library plays the role of the substitution matrix. Moreover, a big difference between COFFEE and Weighted SP is that no extra gap penalties are applied in the COFFEE scheme, since this information is already contained in the library.

COFFEE was the score that TC maximized for generating and evaluating the alignments. However, nowadays TC implements another score that also uses consistency information from the library, called TRIPLET Score.

5.3.2.2 Scoring Functions with Structural Information

This category describes some of the most common scoring functions that incorporate structural sequence information to evaluate the accuracy of the alignments.

- **Root-Mean-Square Deviation score (RMSD):** RMSD is a measure for evaluating the quality of an alignment by using available structural information about the sequences in the alignment [Kab76][Kab78]. RMSD estimates the mean square distance between the equivalent alpha carbons of the two superposed structures.

  In principle, RMSD could also be used as an MSA benchmarking method. It has two advantages over standard methods: no dependence on a reference alignment and the possibility of quantifying the structural correctness of any protein sequence alignments. However, one serious disadvantage is the requirement for a superposition, which is itself a difficult problem. Moreover, other disadvantages are the way that RMSD measures behave with different degrees of sequence divergence and its sensitivity to local or global alignment differences.

- **APDB score:** APDB (Analyze alignments with PDB) is a method proposed by O'Sullivan [OZH+03] to evaluate the quality of an alignment by using available tertiary structures of the sequences in the alignment. It is designed to evaluate how consistent an alignment is with the structural superposition this alignment implies.

  APDB measures the quality of the structural superposition induced by the input alignment given any structures available for the sequences it contains. By treating the alignment as the result of some sort of structural superposition, it simply measures the fraction of aligned residues whose structural neighborhoods are similar.

  Like RMSD, this method can be used as an MSA benchmarking method. In APDB, it is possible to avoid the use of reference alignments when PDB structures are available for at least two homologous sequences in a test alignment. Using this method it should become possible to benchmark or train multiple sequence alignment methods systematically using all known structures, completely automatically. Moreover, APDB solves the major disadvantages of the RMSD score for use as a benchmarking method.

- **IRMSD score:** IRMSD is a redesigned RMSD measure proposed by Notredame in 2006 [AMKN06] to make it independent from any structure superposition procedure. IRMSD is an RMSD based on intra-molecular distance comparisons.

  The iRMSD is a follow up of the APDB measure designed to evaluate alignments for their compatibility with the structural superposition they imply. While APDB was a complex measure depending on three semi arbitrary parameters, the new iRMSD algorithm only requires one parameter.

  Finally, iRMSD was normalized (NiRMSD) in order to take into account the superposition accuracy and the extent of the alignment, making it useful for evaluating the accuracy of structure based sequence alignments.

- **STRIKE score:** STRIKE (Single sTRucture Induced Evaluation), published by Kemena in 2011 [KTKN11], is a method that determines the relative accuracy of two alternative alignments of the same sequences using a single structure.

  Given an alignment between a sequence with a known structure (Template) and a sequence with an unknown structure (Target), the STRIKE score is estimated by projecting all the contacts measured on the Template onto the Target. The scores of these induced contacts are then summed and normalized by the total number of contacts within the Template. The contact score can be further normalized by dividing its value with that of the Template sequence. This measure gives an indication of whether the overall score of the target sequences is significantly lower (<1), comparable (=1) or higher (>1) than that of the only known structure.

5.3.2.3 Multiple Tree Alignment Scoring Functions

As explained in Section 5.2, MTA does not implement any scoring function, but it calls them externally to evaluate the alignments and select the most accurate. In early versions, MTA uses the following single scores to evaluate the alignments: SP, NoRMD, COFFEE, TRIPLET, iRMSD and STRIKE.

Furthermore, MTA has been developed to allow the user to modify any parameters of the metrics used; for example, in the case of the SP score, it is possible to choose the substitution matrix and modify the gap penalty values.

Finally, besides the mentioned metrics, MTA also proposes the use of two meta-scores obtained through genetic algorithms: the Weighted-Score chromosome (WSC) and the Meta-score Code Chromosome (MCC). The main idea of these meta-scores is to combine the main characteristics of various metrics with the goal of improving the evaluation step in order to find more accurate alignments. In the next section, the design of the genetic algorithm, input data used, and genetic algorithm training process to define the two proposed meta-scores are explained in detail.

5.4 Genetic Algorithm Meta-Score

In this section, two heuristic meta-score functions based on a genetic algorithm (GA) are proposed. These algorithms use the information obtained by different quality scores to determine a more realistic method for evaluating the quality of the obtained alignment. Assuming that there is no direct correlation between a scoring metric and the biological quality, we use evolutionary computation (genetic algorithms) from Artificial Intelligence (AI) to search for alternative correlations. To do so, GAs were used to search for the combination of scores that best estimate the biological quality of an alignment. Then, it was analyzed whether this new score could improve the quality of the alignment tools. The approach presented is quite different from such previous ones as SAGA [NH96], which used GAs to build the alignment.

This section presents the design of both GAs, the main aspects of the implementation, the parameter configuration tests to tune the evolution scheme and finally, it presents the GA training process to obtain the two proposed meta-scores.

5.4.1 Genetic Algorithm Design

Genetic algorithms are heuristics based on the principles of natural evolution and survival of the fittest [Hol92]. Solutions are represented by a string encoding, analogous to chromosomes in genetics, and are usually referred to as individuals. The algorithm starts with a number of solutions, the so-called population, which is usually randomly generated. Then, a series of genetic operators (selection, crossover, mutation and replacement) are then applied to the solutions in the population to produce a new population. One full sequence of these operators is called a generation and the process is continued for a number of generations until a stopping criterion is met.

The principle of survival of the fittest guides the search towards good solutions as follows. A suitable target function is used to measure the fitness of each solution. The fitter a solution is, the more likely it is to be chosen in the selection stage to contribute to new solutions. New solutions are formed by the crossover operation, where two of the chosen solutions are selected and recombined to form new solutions. A small proportion of these new solutions are then mutated, i.e. changed slightly in a random way. Once an appropriate number of new solutions have been created, they replace an equivalent number of old solutions, with some of the fitter old solutions surviving to the next generation [Gol89].

To design the GA scheme, it is necessary to define the following structures for each meta-score:

1. The **chromosome** that encodes the solutions space for our optimization problem.

2. The **fitness function** to evaluate the goodness of any solution in the population.

3. The behaviour of the main **evolution operators**, like selection, crossover, mutation or replacement, applied to produce a new population in our scheme.

Next, two proposed meta-score are presented by defining their chromosome, fitness function and evolution operators.

5.4.1.1 Weighted-Score Chromosome

As a first approach, we assume the simplest combination of metrics, which is a weighted scheme among several scoring functions (WSC). Thus, the objective of our chromosome is to select the weight that has to be assigned to each metric in order to maximize the correlation with the biological quality.

In this sense, the chromosome is defined as an array of M genes, M being the number of metrics that we are studying. The *ith* gene of the chromosome is an integer that represents the weight $\in [0, 100]$ of the *ith* metric in the combined scoring function. For example, the chromosome string (100, 0, 0, 50, 25) represents a combined scoring function based on 100% of metric 1, 50% of metric 4 and 25% of metric 5.

The final solution of the GA will inform us about the best weight among all the metrics to achieve the best correlation.

Fitness Function

Ideally, the fitness function has to be able to evaluate the capability of the combined scoring function defined by a chromosome to estimate the biological quality of an alignment. However, it is highly improbable to find a computational scoring function capable of estimating the biological quality quantitatively. Therefore, we opted to search for a scoring function with the power to distinguish qualitatively between good and bad alignments taking the biological quality into consideration. This approach is considered for designing our fitness function. Therefore, the function will evaluate the ability of the meta-score function to select the best biological alignment from a set of solutions derived from aligning the same sequences using different guide trees.

The training is done with several input sequence-sets from the PREFAB database. Multiple alignments are calculated for each input set using the MTA proposal to vary the guide tree. Finally, all the score metrics are calculated for each alignment together with the biological quality score obtained using the Q metric.

This dataset is the foundation of our fitness function. The fitness of each chromosome is calculated by selecting the best alignment for each dataset, which is the one that maximizes the meta-scoring function defined by the chromosome. From the selected alignment, its biological quality (PREFAB Q value) is saved. This procedure is applied to all the input datasets, obtaining the average Q score (i.e. the average biological quality) for the chromosome. Therefore, the goal of the optimization problem is to maximize the average Q value.

Genetic Operators

The proposed WSC GA scheme defines its own mutation operator, while the rest of the operators are the traditional ones. The evolution operators are:

- **Crossover**. Crossover is a reproduction technique that takes two parent chromosomes and produces two child chromosomes. The scheme proposed uses the one-point crossover method. In this method, both parent chromosomes are split into left and right sub-chromosomes, where the left sub-chromosomes of each parent are the same length, as are each parent's right sub-chromosomes. Then each child receives its left sub-chromosome from one parent and the right sub-chromosome from the other. The split position is called the crossover point. For example, if the parent chromosomes are (20,0,15,99,100) and (5,15,25,35,45) and the crossover point is between genes 2 and 3, then the children are (20,0,25,35,45) and (5,15,15,99,100).

- **Selection**. The selection operator is used to choose which individuals should mate. The selection proposed method picks an individual based on its fitness score compared to the rest of the population. The higher the score, the more likely an individual is to be selected. Any individual has a probability s of being chosen, where s is equal to the fitness of the individual divided by the sum of the fitnesses of each individual in the population.

- **Mutation**. Mutation is a common reproduction operator used to find new points to evaluate in the search space. When a chromosome is chosen for mutation, a random choice is made of some of its genes, and these are modified. In the case presented, we work with the following two approaches:

  - Bit-level Mutation, where all the bits in the chromosome string can mutate individually. In this case, the mutation means that the corresponding bit is flipped (from 0 to 1 or from 1 to 0), changing the value of the corresponding weight.
  - Gene-level Mutation, where the whole gene, *ith* metric weight, changes if the mutation takes place. To change the genes, a random number between 0 and 100 is generated.

  In both cases, a defined mutation probability p is applied to each mutable element (bit or gene). Then, the algorithm randomly decides if the mutation happens and, if so, it changes the element.

- **Replacement**. The replacement policy depends on the selection and mutation scheme. Each generation creates a new population of individuals by selecting from the previous population then mating to produce offspring for the new population. Moreover, the best chromosomes from each generation are carried over to the next generation to accelerate the convergence.

5.4.1.2 Meta-Score Code Chromosome

The previous WSC Chromosome have some limitations because they are unable to establish relations between scores and other metrics in the sense of "*if score 1 is bigger than the average of score 1 then it is a good alignment.*". This limits the improving capacity of the genetic algorithm. We therefore opted to design a more complex and promising evolution scheme based on calculating the meta-score through a program code that can evolve to maximize the fitness function.

A brief language for coding the chromosome programs (MCC) is defined. This grammar is shown in Algorithm 5 in BNF format. Basically, the meta-score program is composed of several conditional sentences with the corresponding assignment statements. The conditionals define the relations between the score metrics, score statistics and numeric constants that must be satisfied in order to execute the statement. The statements modify the METASCORE variable through different arithmetical and assignation operations. Algorithm 6 shows an example of these MCC programs.

Algorithm 5 BNF grammar of chromosome code

```
< pgm >::= {< sentece >} ’END’
< sentence >::=< if > | < if >< sentence >
< if >::=< enabled > ’IF’ ’(’ < condition > ’)’ ’THEN’ < statement >
< enabled >::= ’On’ | ’Off’
< condition >::=< term > {< opr_comp >< term >}
< term >::=< score > | < statistics > | < NUMBER >
< score >::= Score1 | Score2 | .. | ScoreN
< statistics >::= Statistc1 | Statistc2 | .. | StatistcN
< opr_comp >::= ’==’ | ’!=’ | ’>=’ | ’<=’
< opr_cond >::= ’&&’ | ’||’
< statement >::= ’METASCORE’ < opr_asig >< term > {< opr_arit >< term >} ’;’
< opr_asig >::= ’==’ | ’+=’ | ’>=’ | ’<=’
< opr_arit >::= ’+’ | ’-’ | ’*’ | ’/’ | ’;’
```

Algorithm 6 Meta-score code example

```
On If (Score1>50 && Score2>50 && Score3>50 && Score4>50) then METASCORE += 100;
On If (Score1<50 && Score2<50 && Score3<50 && Score4<50) then METASCORE -= 100;
Off If ( Score1>GlobalMean || Score2>GlobalMean) then METASCORE += Score1 + Score2;
On If (Score1==GlobalMax && Score1!=GlobalMin) then METASCORE += Score1 * 2;
Off If (Score1>MeanScore1) then METASCORE += Score1;
On If (Score2>MeanScore2) then METASCORE += Score2;
On If (Score3>MeanScore3) then METASCORE += Score3;
END
```

Each chromosome is a program coded with this language. This code can be mutated and reproduced during the evolution. The program is executed by its own parser each time that the fitness function needs to evaluate an alignment. As a result of the execution, the program returns a representative meta-score of the quality of the alignment. To delimit the solution search space, the program bytecode size is limited to only 256 bytes. However, this is big enough to obtain a good meta-score metric.

MCC introduces several changes performed on the genetic to deal with the new chromosome. The fitness function is the same as for the first scheme. The only difference is that the meta-score is not obtained through a weighting of different scores, but by executing the chromosome program.

Genetic Operators

The selection and replacement operators have not changed from WSC version. However, the crossover and mutation operators need to be modified. These operators work with the program bytecode, respecting the following rules:

1. Changes produced in the code have to be syntactically and semantically correct, in order for the resulting program to be executed without errors.

2. The size of the programs has to be controlled to avoid unlimited growth.

The behavior of the modified operators is the following:

- **Crossover**. The program merging is only done at the level of full sentences. We use a two-point crossover method to choose a fragment of the chromosome A program and a fragment of the chromosome B code that will be exchanged to create the two child chromosomes. The fragment of code is defined by selecting the starting and ending sentence lines randomly.

- **Mutation**. In order to guarantee the syntax of the new program, the mutation operations are only performed with certain symbols and respecting the constraints defined by the grammar. Some symbols, like '*end*' of program, '*then*' or '*;*' cannot be changed.

5.4.2 Genetic Algorithm implementation

The MCC and WSC GAs are implemented using a specialized library called GAlib [Wal96]. GAlib is a C++ library of GAs components that includes tools for use to develop any genome using any representation and genetic operators.

For the GA training, it was first necessary to generate all the input data. The input data is a list of n alignments of a range of sequence sets evaluated with different metrics. Some PREFAB sequence sets were used to build it. Each sequence set was an experiment and these were aligned using MTA-CLW to obtain n alignments for each experiment. Then, the n alignments were evaluated using different score metrics. These scores are used by the GA to evaluate the goodness of each chromosome meta-score and develop it to find a good combination between them. The metrics used as input data were NorMD, COFFEE, TRIPLET, STRIKE, NiRMSD and the following three variations of SP score found in the literature:

- An SP score using BLOSUM62 as a substitution matrix with a gap-opening penalty of -11 and a gap-extension penalty of -1 (SP B62-1).
- An SP score using BLOSUM62 as a substitution matrix with a gap-opening penalty of -6.6 and a gap-extension penalty of -0.9 (SP B62-2).
- An SP score using PAM250 as a substitution matrix with a gap-opening penalty of -13.8 and a gap-extension penalty of -0.2 (SP G250).

The next step was to implement the chromosome designed in Section 5.4.1, which represents the population, to evolve and define the fitness function. The proposed fitness function, where $g(i)$ is a gene i and SC_i is an input score, evaluates each chromosome for all the input score list in each experiment and tries to obtain the maximum value that returns a Q score. Finally, the average Q score from all the experiments is obtained.

The GA is started by initializing each chromosome of the initial population. Then, the evolution is performed over successive generations until the fitness score converges. The convergence consists of defining a percentage c and the number g of previous generations to compare against. The convergence stops the evolution when the best-score during the last g generations is within c of the current generation's best individual score.

5.4.3 Genetic Algorithm training process

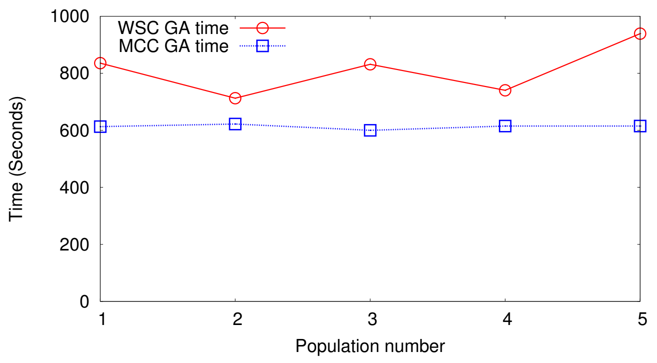
A GA must be trained with a specific data set to find the fittest new scoring functions. Before that, the GAs must be configured to tune all the parameters and thus find the improved fittest solution. This section shows the experiments focused on setting the GA implementations. It then presents the evolution process and the two meta-scores obtained from running the refined GAs.

The MSA PREFAB benchmark was used to provide sets of sequences to use as experiments for running the GAs and analysing the biological quality of the alignments using the Q score. In order to avoid the over-fitting problem, the validation methodology used two disjointed data sets. Although both used the PREFAB benchmark, the experiments used for GA training differed from those used in further sections to validate MTA with the proposed meta-scores. For the training process, 100 of 1682 sequence sets (experiments) were used, and for each experiment, 100 alignments were evaluated. The configuration and training process were run on a Intel(R) Xeon(R) E5462 2.80 GHz with 8 cores and 16 GB of RAM.

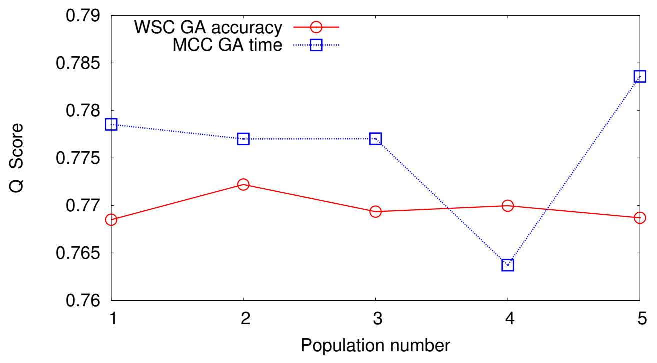
5.4.3.1 Genetic Algorithm configuration

The GA setting study attempted to determine the best parameter configuration for both the WSC and MCC algorithms in order to balance the algorithm execution time and the obtained accuracy. Thus, the parameters to be studied were: the population number, the population size, the mutation probability and finally the crossover probability. The results obtained from this study are applied to find the two meta-scores used by MTA to evaluate alignments and find the most accurate one.

The first parameter, population number, is evaluated in Figures 5.3a and 5.3b in the range from 1 to 5. For all the tests, it can be seen that the execution time for the MCC algorithm remained stable while WSC varied from 700 to 900 seconds. Observing the accuracy analysis, the best results for the MCC came from the population number equal to 2. For WSC the best accuracy was obtained with 5 populations, but at the cost of increasing the execution time by 32%. Therefore, two populations were adopted as a good compromise between accuracy and execution time for the WSC GA.



(a) Execution time varying the population number



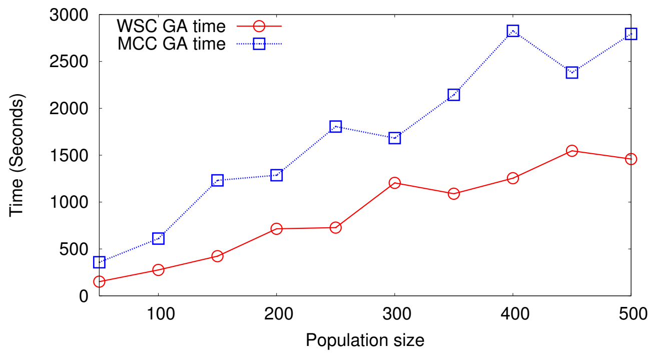
(b) Accuracy analysis varying the population number

Figure 5.3: Genetic Algorithm configuration study: Population number

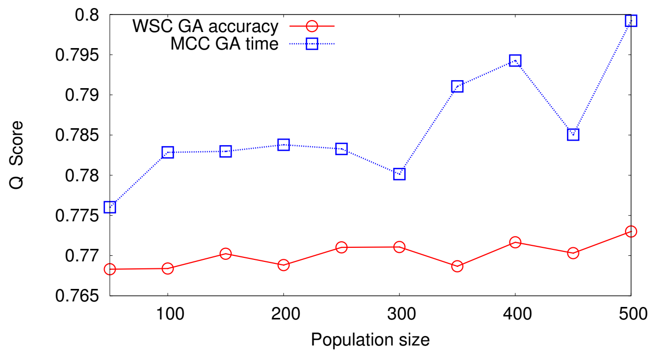
The population size parameter was evaluated in the range from 50 to 500, as shown in Figures 5.4a and 5.4b. The execution time and alignment accuracy increased with the population size in both WSC and MCC, and with a population size of 500, the alignment accuracy was good enough to choose these values in further executions.

The next parameter, the mutation probability was expanded from 0.01 to 0.1. In this case, both WSC and MCC execution times increased with the mutation probability grows. Thus, the accuracy results were obtained with a probability of 0.09 for WSC and 0.03 for MCC.

Finally, Figures 5.6a and 5.6b show the crossover probability study, from 0.1 to 1. For WSC, the runtime grew slightly when the crossover probability increased, while the runtime behaviour for the MCC was irregular. In this case, the best quality result was with a probability of 0.9 for WSC and 0.4 for MCC.

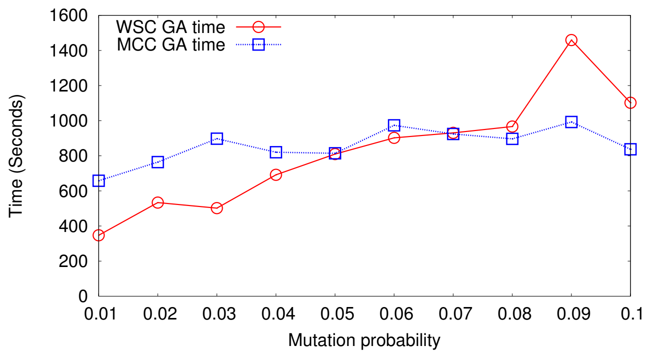


(a) Execution time varying the population size

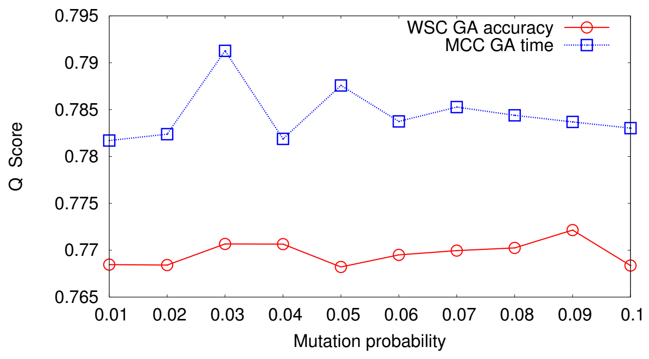


(b) Accuracy analysis varying the population size

Figure 5.4: Genetic Algorithm configuration study: Population size



(a) Execution time varying the mutation probability



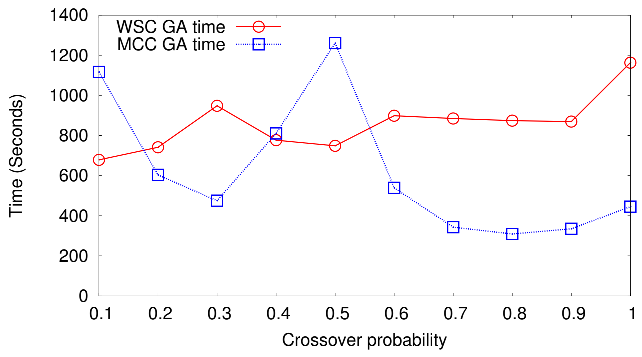
(b) Accuracy analysis varying the mutation probability

Figure 5.5: Genetic Algorithm configuration study: Mutation probability

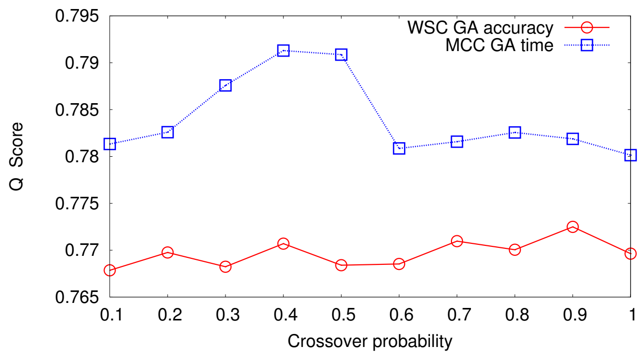
Table 5.1 shows the configuration values obtained for both WSC and MCC. As a conclusion, we can state that a good configuration for future tests for WSC is to use 2 populations of 500 individuals with a mutation probability of 9% and a crossover probability of 90%. The best set-up for MCC is to use 2 populations of 500 individuals with a mutation probability of 3% and a crossover probability of 40%.

Parameter	**WSC**	**MCC**
Population number	2	2
Population size	500	500
Mutation probability	9%	3%
Crossover probability	90%	40%

Table 5.1: WSC and MCC configuration parameters



(a) Execution time varying the crossover probability



(b) Accuracy analysis varying the crossover probability

Figure 5.6: Genetic Algorithm configuration study: Crossover probability

5.4.3.2 Genetic Algorithm evolution

After configuring the genetic algorithm for both WSC and MCC, the two settings obtained were then used to evolve both meta-score schemes. The GA evolution was done using the same hardware infrastructure and training data sets as in previous experiments.

Figure 5.7 shows the evolution process for both GA proposals. It can be seen how the biological quality achieved using WSC and MCC was enhanced in function of the number of GA generations. In both schemes, the quality improved rapidly in the early generations. For WSC, note that it reached its maximum quality of 0.774 after only 150 generations. In comparison, MCC required 1,500 generations to achieve its best quality because its search-space was bigger than that of WSC. However, this higher complexity of MCC was

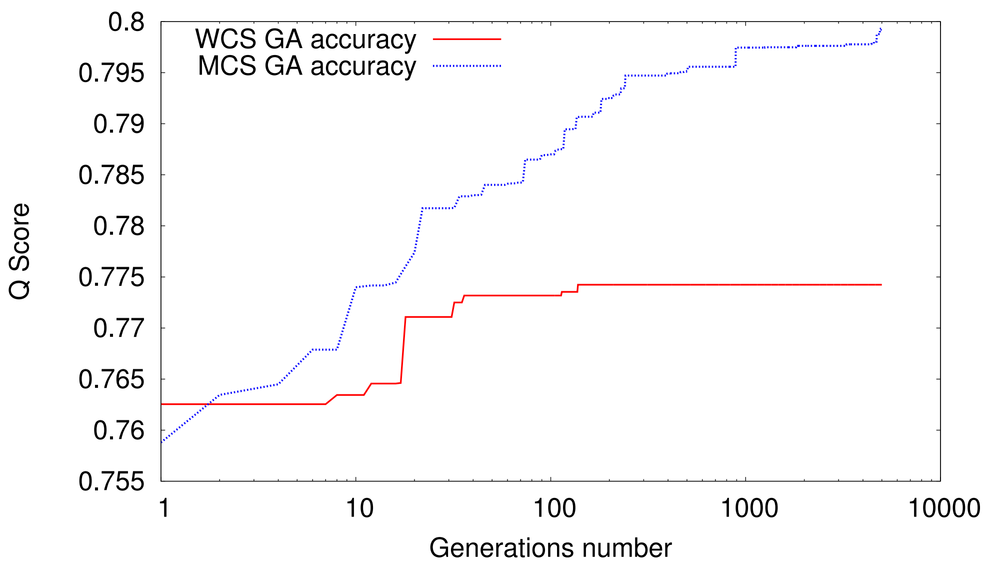


Figure 5.7: MCC and WSC GA accuracy evolution study

offset by better quality results. MCC quality increased up to 0.799, as it was able to identify more accurate alignments than WSC.

Finally, after the evolution process, the resulting GA meta-scores for both WSC and MCC were:

- **WSC meta-score:** The GA score weights obtained running WSC GA were (1, 14, 1, 0, 37, 1, 98). These weights represent a combined scoring function based on 1% of SP B62-1, 14% of SP B62-2, 1% of SP G250, 37% of STRIKE, 1% of TRIPLET and 98% of NiRMSD. These results show that more weight was given to both structural scores (NiRMSD and STRIKE) because they were more sensitive.

- **MCC meta-score:** The MCC results were achieved using the meta-score function obtained from the GA evolution. This resulting program code is shown in Algorithm 7. This algorithm shows all the resulting conditionals that define the relations between the score metrics and have to be satisfied to calculate the METASCORE variable used to evaluate the alignments. In the code, Score1 corresponds to SP B62-1, Score2 to SP B62-2, Score3 to SP G250, Score4 to NorMD, Score5 to STRIKE,

Algorithm 7 Meta-score code program

```
**On if (Score6 > 105 || Score7 > Score1 || MaxScore1 > 212 || Score7 > Score7 || 214 > MinScore3) then METASCORE *= MaxScore2;**
**On if (MaxScore4 > Score6 && MaxScore7 < Score1 && MeanScore4 < 242 || Score3 ! = MaxGlobal || 227 != Score3) then METASCORE += Score5;**
On if (36 > Score4 || Score3 > 37) then METASCORE += MinGlobal + Score2;
On if (Score3 == Score4 || Score4 > 103) then METASCORE /= 78 / MaxScore4;
**On if (Score2 < 29) then METASCORE /= Score3;**
On if (Score2 ! = 97) then METASCORE += Score7;
On if (Score1 ! = 91) then METASCORE += 246;
On if (Score5 ! = MeanScore4) then METASCORE += MaxScore2;
On if (Score2 ! = MeanScore4) then METASCORE += MeanScore5;
On if (Score1 ! = 6) then METASCORE -= 26;
Off If (133 ! = MaxScore3) then METASCORE /= 149;
On if (MinScore3 < Score4) then METASCORE += MaxScore2;
Off If (Score7 ! = MeanGlobal) then METASCORE /= MinScore4;
On if (MeanScore5 ! = 71) then METASCORE -= Score4;
**On if (Score3 > 10) then METASCORE -= 33;**
END
```

Score6 to TRIPLET and Score7 to NiRMSD. Of course, there are sentences that have no effect on the METASCORE value (for example, sentences 11 and 13) and only a few lines are significant (in bold in the algorithm). For instance, line 1 increases the meta-score if the NiRMSD score is greater than the SP B62-1. The last instruction decreases the meta-score if the SP G250 score is lower than 10%. The most significant conditionals, which are highlighted in bold, contribute to modifying and finally obtaining a METASCORE, which, by combining the properties of the compared metrics, is capable of evaluating alignments and obtaining results closer to the biological quality.

5.5 Multiple Tree Alignment Experimentation

This section presents the experimental results focused on first validating the proposed GA meta-scores, comparing them to the other existing metrics, and then analyzing the accuracy and scalability improvements of the proposed method, MTA. To this end, the following experiments were carried out:

- In the first study, presented in Section 5.5.1, the biological quality of the alignments chosen with the new GA meta-scores were compared with the quality of the same alignments chosen using other existing metrics.

- The second study, presented in Section 5.5.2, shows the accuracy of the alignments obtained by MTA proposal and it compares them against other MSA programs found in the literature.
- In the third study, presented in Section 5.5.3, the performance and scalability of the parallel version of MTA is analyzed.

5.5.1 Comparison of the Evaluation Metrics

This experiment evaluated the ability of standard metrics (SP, NoRMD, STRIKE, COFFEE, TRIPLET, NiRMSD) and the proposed evaluation metrics (WSC and MCC) to choose good alignments, identify which score was able to make the best selection and finally compare them against the original MSA method alignment.

The experiment was done using two MTA configurations fixing the MSA program and varying the evaluation metric (Tables 5.2, 5.3). The MSA applications used by MTA were ClustalW (MTA-CLW) and T-Coffee (MTA-TC), the number of evaluated trees being 300 for MTA-CLW and 100 for MTA-TC. The input metrics compared were SP, NoRMD, STRIKE, COFFEE, TRIPLET, NiRMSD and the proposed meta-scores WSC and MCC. The input datasets used were the whole PREFAB benchmark sequence sets divided into five groups according to the percent identity of the sequences. The accuracy results were obtained by using PREFAB Q score comparing the resulting alignments with the reference ones. In each table, the columns identify the range of identity, the rows identify the evaluation metric selected, the first one being the result for the original aligners, and the second one, in red, the optimal accuracy result using PREFAB Q score as the selection metric.

Table 5.2 is related to the MTA-CLW configurations. As can be observed, MTA-CLW MCC obtained the highest average accuracy in each PREFAB category, except in the range of 35-100, where MTA-CLW NiRMSD was the most accurate. In total, MTA-CLW MCC was the best configuration, being 15.40% more accurate than the original MSA method, ClustalW (0.712 vs 0.617). In spite of the improvement, there were further potential gains until the optimal quality was reached (MTC-CLW Q row). Regarding the WSC meta-

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**
ClustalW	0.289	0.605	0.816	0.941	0.617
MT-CLW Q	0.552	0.787	0.908	0.976	0.779
MTA-CLW SP	0.345	0.637	0.837	0.953	0.649
MTA-CLW NorMD	0.350	0.641	0.826	0.950	0.649
MTA-CLW STRIKE	0.392	0.688	0.857	0.954	0.686
MTA-CLW TRIPLET	0.369	0.672	0.850	0.949	0.670
MTA-CLW COFFEE	0.369	0.673	0.852	0.954	0.671
MTA-CLW NiRMSD	0.422	0.711	0.870	**0.957**	0.705
MTA-CLW WSC	0.427	0.718	0.870	0.950	0.708
MTA-CLW MCC	**0.434**	**0.720**	**0.875**	0.950	**0.712**

Table 5.2: Comparison between different MTA-CLW configurations varying the evaluation metric

score, MT-CLW WSC was the second most accurate configuration, improving the alignment accuracy of ClustalW by 14.75% (0.708 vs 0.617). If we do not take the GA meta-scores into consideration, the best score was NiRMSD that surpassed the remaining single scores.

The results obtained for the MTA-TC are shown in Table 5.3, these being on average better for the MTA-TC configurations, and more accurate than the alignments built with TC. In total, the MTA-TC MCC and MTA-TC NiRMSD configurations were the most accurate, improving the alignment quality of T-Coffee by 3.10% (0.731 vs. 0.709). It is seen that the MCC total average accuracy result was the same as the one for NiRMSD one. Moreover, the WSC accuracy results were very close to those from NiRMSD (0.730 vs. 0.731).

From Table 5.3, it can also be deduced that, regarding T-Coffee, improvements in MTA-TC were less significant than those in MTA-CLW regarding ClustalW. This is because T-Coffee introduces consistency information into the alignments, correcting the errors caused by the guide tree in progressive alignments. Also, we can observe that on average GA meta-scores were unable to achieve better results than those for NiRMSD. This is due to the NiRMSD score clearly surpassing all the remaining scores. Therefore, the meta-score cannot provide a better quality than NiRMSD alone.

The above results prove that the best evaluation score is the MCC heuristic. This is a consequence of MCC being a meta-score obtained by GA training

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**
T-Coffee	0.421	0.721	0.876	0.951	0.709
MT-CLW Q	0.509	0.778	0.902	0.959	0.759
MTA-TC SP	0.427	0.722	0.876	**0.953**	0.711
MTA-TC NorMD	0.430	0.726	0.875	**0.953**	0.714
MTA-TC STRIKE	0.435	0.726	0.875	0.950	0.715
MTA-TC TRIPLET	0.430	0.723	0.877	0.950	0.712
MTA-TC COFFEE	0.426	0.725	0.875	**0.953**	0.712
MTA-TC NiRMSD	**0.470**	0.743	0.885	**0.953**	**0.731**
MTA-TC WSC	0.457	**0.748**	0.888	0.944	0.730
MTA-TC MCC	0.459	**0.748**	**0.889**	0.944	**0.731**

Table 5.3: Comparison between different MTA-TC configurations varying the evaluation metric

and it consists of a program code that combines the properties of different evaluation metrics.

It is important to note that, with MCC-CLW and MCC-TC, all MTA configurations produce a significant improvement in the alignment quality, independently of the score selected to evaluate the multiple alignments. It is also remarkable, that scores with structural information like STRIKE (0.686 and 0.715) and NiRMSD (0.705 and 0.731) obtain better results than traditional scores like COFFEE (0.671 and 0.712).

Another MTA feature is that the error associated with the guide tree is bigger when the sequences are less correlated. This effect can be validated with the percentage of improvement achieved in the different PREFAB dataset categories. With low related sequences (0-15 column) the improvement reaches 50% with ClustalW and 11% with T-Coffee. However, with high correlated sequences (the remaining columns) the improvement is less than 20% and 3% respectively.

5.5.2 Alignment Accuracy analysis

Table 5.4 presents the comparison of alignment accuracy between the proposed method using ClustalW and T-Coffee, which obtained the best configurations in subsection 5.5.1 (MTA-CLW MCC, MTA-TC MCC, MTA-TC

NiRMSD), and some of the most common consistency-based, iterative or progressive alignment MSA applications found in literature (MSAProbs [LSM10], MAFFT [KMKM02], Probalign [RULD06], ProbCons [DMBB05], ClustalΩ [SWD+11], Muscle [Rob04], Dialign-tx [SHS+10] and FSA [BRS+09]). All the MSA methods were run using the default parameters. The experiment was carried out using the whole PREFAB benchmark sequence sets divided into five groups according to the percent identity of the sequences. The accuracy results were obtained by using the PREFAB Q score. In the table, the columns identify the range of identity and the rows identify the method compared. The last column indicates whether the method used consistency or not.

The results show that the consistency-based methods (MSAProbs, MAFFT, Probalign, MTA-TC MCC, MTA-TC NiRMSD, ProbCons and T-Coffee) are the most accurate. This is because consistency-based methods incorporate a larger share of information into the evaluation. However, the construction of this information increases the execution time for these methods, limiting them reduced sets of alignments. As it is seen in Subsection 5.5.1, MTA-TC MCC and MTA-TC NiRMSD improve the alignments accuracy of T-Coffee by 3.10% . However, they are also more accurate than ProbCons by 2.09%, Probalign by 1.67% and MAFFT by 0.97%; becoming the second and third most accurate methods. Moreover, it can be noticed that when sequences are not highly correlated (the first column in Table 5.4), MTA-TC NiRMSD is able to improve the quality of MSAProbs by 3.52%, it being the best method in such conditions. Regarding MTA-TC MCC, the alignments of this first column are 1.1% more accurate than the ones obtained with MSAProbs. This feature is especially important for large-scale alignments (ten thousands of sequences) where the correlation of the sequences is lower.

On the other hand, the MTA-CLW MCC approach shows great improvements since not only is it more accurate than all non-consistency methods, but also it is able to overcome the consistency-based method, namely T-Coffee. MCC-CLW MCC improves the alignment quality of the original ClustalW by 15.40%, Dialign-tx by 12.22%, Muscle by 5.17%, ClustalΩ by 1.71% and T-Coffee by 0.42%. This converts it into the seventh most accurate method, although it does not introduce consistency information.

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**	**Con**
MSAProbs	0.454	**0.756**	**0.900**	0.961	**0.738**	Yes
MTA-TC MCC	0.459	0.748	0.889	0.944	0.731	Yes
MTA-TC NiRMSD	**0.470**	0.743	0.885	0.953	0.731	Yes
MAFFT	0.431	0.743	0.887	0.958	0.724	Yes
Probalign	0.424	0.732	0.891	0.962	0.719	Yes
ProbCons	0.425	0.729	0.888	0.956	0.716	Yes
MTA-CLW MCC	0.434	0.720	0.875	0.950	0.712	No
T-Coffee	0.421	0.721	0.876	0.951	0.709	Yes
ClustalΩ	0.395	0.708	0.878	**0.965**	0.700	No
Muscle	0.365	0.684	0.860	0.951	0.677	No
Dialign-tx	0.290	0.617	0.824	0.955	0.625	No
ClustalW	0.289	0.605	0.816	0.941	0.617	No
FSA	0.110	0.467	0.805	0.957	0.516	No

Table 5.4: Comparison of the accuracy of MSA methods

There is the appraisal that the consistency effect, obtained with a cost in time and memory requirements, can be partially solved by the correct construction of the guide tree. Moreover, we believe that, over recent years, the impact of the guide tree over the alignment quality has been underestimated.

Overall, our proposed methods are not only more accurate than T-Coffee and ClustalW, but also more accurate than other MSA aligners. To date, two versions have been developed: one with a consistent method (TC) to produce more accurate alignments, and the other, faster (CLW) to build larger alignments. However, this method can be implemented using any MSA aligner that accepts any guide tree as an input parameter. Furthermore, in line with Figure 5.2, these results can be improved by developing an evaluation function capable of finding the best guide tree.

5.5.3 Scalability study

This experiment presents a scalability study to compare the running time of our proposed methods against the running time of the applications used to align the CLW and TC trees (Figure 5.8a, 5.8b). Naturally, the serial version of our proposals cannot compete with other MSA aligners on executing time, because the trees have to be aligned serially using the aligners. However, owing to the fact that each individual alignment is independent and can be done separately from the others, MTA was developed to be capable of aligning the alignments in parallel on a distributed system. Therefore, Figures 5.8a and 5.8b in this section show the scalability study of the parallel version implemented with MPI and run on a cluster. The cluster used was made up of 24 computing nodes. Each computing node contained a 2,4 GHz Intel Core 2 Quad and 8GB of RAM, giving a total of 96 cores.

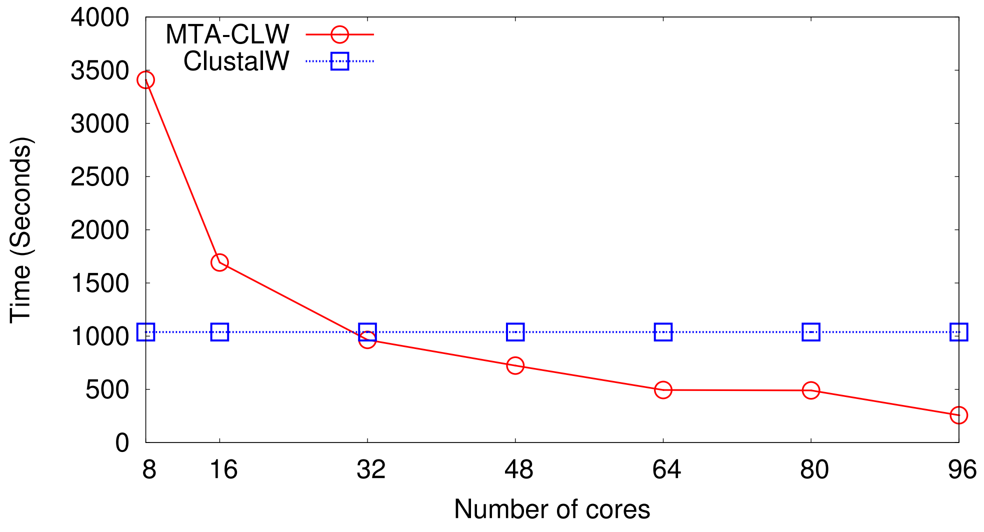
Two prefabricated sets of sequences from the Pfam database were used to do this experiment. A bigger one was made up of 1,000 sequences to test MTA-CLW and a smaller one of 200 sequences to test MTA-TC, because TC is not capable of aligning large sets of sequences.

The parallel implementation followed the master-worker paradigm, where 1 task is responsible for creating the N guide trees and also selecting the best alignment; and N tasks that perform the alignment in parallel. Thus we require a total of $N + 1$ tasks, N being the number of different guide trees we want to align. In the experimentation we used a cluster with 96 computation nodes, therefore, we could align up to N=95 guide tress in parallel, which is the optimal configuration in such infrastructure.

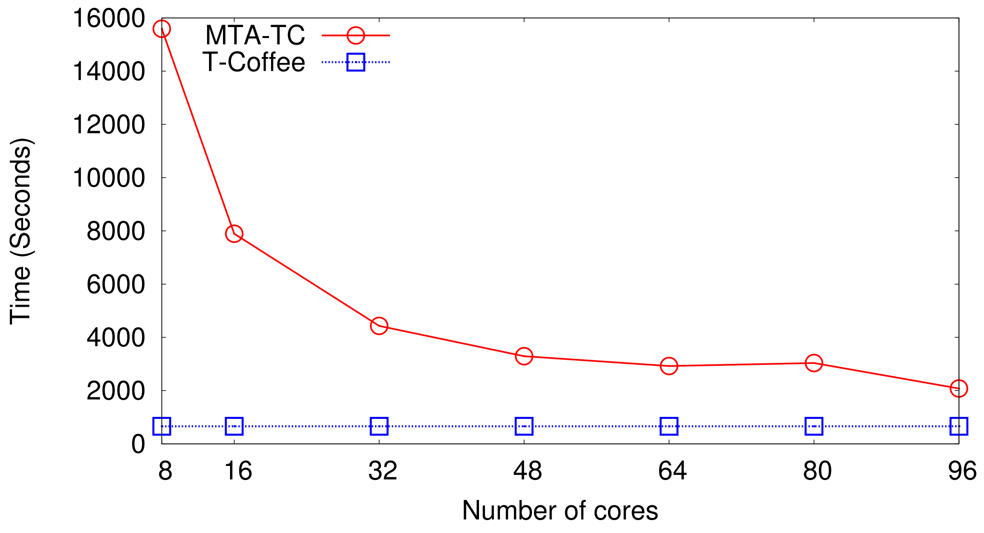
Figure 5.8a shows the runtime comparison of MTA-CLW with the number of cores increasing from 8 to 96. First of all, the parallel version of MTA-CLW is a scalable method capable of reducing the running time by 92% when the number of cores increases. With 32 cores, MTA-CLW has already improved the CLW runtime, and in the best case (96 cores), MTA-CLW is 75% faster than CLW. The reason why our proposal is faster than CLW is because the CLW distance matrix is built from full dynamic programming alignments using two gap penalties and a full amino acid weight matrix, while our proposal is built from the statistical ktup method, which is less sensitive but faster.

Figure 5.8b, related to MTA-TC, shows that it is capable of reducing the running time by 87% when the number of nodes increases from 8 to 96. In this case, MTA-TC does not improve the TC, and in the best case, MTA-TC is 68% slower than TC. These results are due to the time penalty introduced by the consistency scheme. In the case of TC, consistency is achieved through a library of alignment, so the increase in runtime is due to the master having to create this library and then individual workers having to read it to obtain the alignments.

As a conclusion, the MTA proposals are scalable as the number of processors increases, as long as the number of processors is less than $N-1$. Moreover, parallel MTA is generally able to produce alignments in a similar time as the original MSA methods.



(a) MTA-CLW execution time with an increasing number of processors



(b) MTA-TC execution time with an increasing number of processors

Figure 5.8: Parallel MTA Scalability study

5.6 Integration of Consistency Library Optimization with MTA

In this section, we present the integration study of the consistency library optimization approach with the MTA method. This consists of aligning the multiple guide trees with T-Coffee (TC) using the consistency library optimization method presented in Chapter 4. One objective is to compensate the loss in alignment quality caused by the reduction in consistency information by aligning a more accurate tree. Furthermore, another goal is to reduce the execution time of MTA-TC using memory optimization approaches.

To study the behavior of MTA-TC with the optimized library, some of the experiments, shown in Chapter 4 and 5, were repeated. Specifically, the following experiments were carried out:

- The first study, presented in Section 5.6.1, compared the alignment accuracy of different MTA-TC using different consistency library optimization approaches in order to select the most appropriate.
- The second study, presented in Section 5.6.2, showed the accuracy of the alignments obtained by MTA-TC using the consistency library optimization approaches, which were chosen in the first study, and these alignments were compared with other MSA programs found in the literature.
- In the third study, presented in Section 5.5.3, the performance and scalability of the sequential and the parallel version of optimized MTA were analyzed.

5.6.1 Consistency Optimization configuration

This experiment evaluated MTA-TC alignments using different consistency optimization configurations in order to find a TC library optimization that provides a good compromise between time and alignment accuracy.

The experiment was done using MTA-TC and the Threshold-Library optimization approach (T-Library) varying the threshold value from 10% to 80%.

Table 5.5 shows the comparison of the alignment quality of the different MTA-TC optimization approaches with MTA-TC without consistency information and the standard TC. Furthermore, the table also shows the comparison between the sum of the total execution time taken by the optimized TC to align the best alignments and the execution time of TC. The input metric for selecting the best alignment was NiRMSD and 100 guide trees were analyzed. The input datasets, used to validate the accuracy, were only fourteen PREFAB benchmark sequence sets divided into five groups according to the percent identity of the sequences. In the table, the first four columns identify the range of identity, and the last column identifies the total execution times. Otherwise, the rows identify the different T-Library approaches selected, the first one being the result for the standard TC, and the second one, the MTA-TC without library optimization.

The results, shown in Table 5.5, determine that, with only fourteen sequence sets from PREFAB database, MTA-TC T-Library10 is the best approach because it was able to increase the alignment accuracy of TC by 5.98% while reducing the execution time by 26.12%. Furthermore, it did not lose accuracy compared with MTA-TC without optimization, but increased it by 0.12%, because if the quality of consistency information is sometimes poor, it may worsen the quality of the alignment. It can also be seen that the alignment accuracy and the execution time are slightly reduced as the consistency optimization increases. Specifically, since T-Library70, the alignment accuracy of MTA-TC is worse than the standard TC. It is also assumed that the

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**	**Time (s)**
TC	0.235	0.833	0.896	0.981	0.736	1646
MTA-TC NiRMSD	0.394	0.846	0.896	0.981	0.779	1646
MTA-TC T-Library10	0.402	0.850	0.886	0.982	0.780	1216
MTA-TC T-Library20	0.385	0.851	0.886	0.982	0.776	1210
MTA-TC T-Library30	0.365	0.845	0.883	0.982	0.769	1198
MTA-TC T-Library40	0.336	0.840	0.879	0.983	0.759	1190
MTA-TC T-Library50	0.319	0.840	0.878	0.983	0.755	1184
MTA-TC T-Library60	0.337	0.817	0.871	0.981	0.752	1176
MTA-TC T-Library70	0.238	0.826	0.879	0.981	0.731	1171
MTA-TC T-Library80	0.179	0.824	0.873	0.978	0.713	1163

Table 5.5: Comparison between different MTA-TC T-Library configurations varying the level of optimization

small reduction in time as the optimization threshold increases is because the prefab sequence sets are small, and this difference will increase with large sets of sequences

According to these results, the MTA-TC T-Library10 was chosen for further experiments. However, in some experiments, the MTA-TC T-Library50 was also used to compare to T-Library10 because its alignments are also better than the standard TC alignments and it provides a higher memory optimization scheme.

5.6.2 Alignment Accuracy analysis

Table 5.6 shows the same experiment as the one shown in Table 5.4, but in this case it presents a comparison of the alignment accuracy between MTA-TC T-Library10 and T-Library50 using NiRMSD as a selection metric, with some of the most common consistency-based, iterative or progressive alignment MSA applications. The experiment was done using the whole PREFAB benchmark sequence sets divided into five groups according to the percent identity of the sequences. The accuracy results were obtained by using the PREFAB Q score. The columns in the table identify the range of identity and the rows identify the method compared. The last column indicates whether the method used consistency or not.

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**	**Con**
MSAProbs	0.454	**0.756**	**0.900**	0.961	**0.738**	Yes
MTA-TC NiRMSD	**0.470**	0.743	0.885	0.953	0.731	Yes
MAFFT	0.431	0.743	0.887	0.958	0.724	Yes
MTA-TC T-Library10 NiRMSD	0.449	0.733	0.878	0.946	0.720	Yes
Probalign	0.424	0.732	0.891	0.962	0.719	Yes
ProbCons	0.425	0.729	0.888	0.956	0.716	Yes
MTA-CLW MCC	0.434	0.720	0.875	0.950	0.712	No
MTA-TC T-Library50 NiRMSD	0.428	0.723	0.877	0.948	0.711	Yes
T-Coffee	0.421	0.721	0.876	0.951	0.709	Yes
ClustalΩ	0.395	0.708	0.878	**0.965**	0.700	No
Muscle	0.365	0.684	0.860	0.951	0.677	No
Dialign-tx	0.290	0.617	0.824	0.955	0.625	No
ClustalW	0.289	0.605	0.816	0.941	0.617	No
FSA	0.110	0.467	0.805	0.957	0.516	No

Table 5.6: Comparison of the accuracy of optimized MTA-TC approaches with other MSA methods

The results show that although the alignment accuracy of MTA-TC T-Library10 is 1.5% worse than the same method without consistency optimization (MTA-TC), its quality is still better than the standard TC, and is the fourth most accurate method among the ones compared. Specifically, the MT-TC T-Library10 is 1.55% more accurate than TC.

Regarding the MTA-TC T-Library50, as expected and due to the further memory optimization, it is 2.74% less accurate than MTA-TC and 1.25% than the MTA-TC T-Library10. However, its accuracy is still better than the standard TC by 0.28%, and is thus the eighth most accurate method.

5.6.3 Scalability study

In this experiment, the scalability study of the integration of the consistency optimization scheme with MTA-TC was analyzed and compared with standard MTA-TC and TC. This experiment was divided into two, the first one to analyze the performance of the parallel version of MTA-TC integrating the consistency library optimization approach and the second one, to study the sequential version of memory optimized MTA-TC and compare it with TC.

5.6.3.1 Parallel Multiple Tree Alignment

This section presents the scalability study of the parallel version of MTA-TC using the consistency library optimization scheme. To achieve this, the scalability studies, presented in Sections 4.4.2 and 5.5.3, were repeated to analise the reduction in the memory requirements and execution time, and the improvements in the scalability of parallel MTA-TC.

Figure 5.9 show the same experiment as Figure 5.8b, but it presents the runtime comparison of parallel MTA-TC T-Library10 and T-Library50 against MTA-TC and TC, increasing the number of cores from 8 to 96. This experiment was done using a sequence set made up of 200 sequences and it was run on a cluster made up of 24 computing nodes. Each computing node contained a 2.4 GHz Intel Core 2 Quad and 8GB of RAM, giving a total of 96 cores. In this experiment, 95 guide trees were analyzed because the cluster was composed of 96 cores, this being the optimal configuration in such infrastructure.

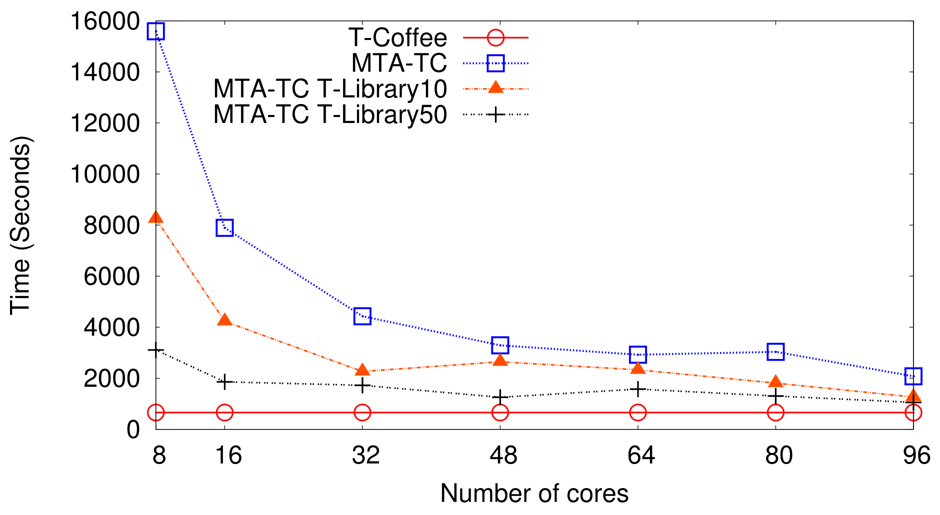
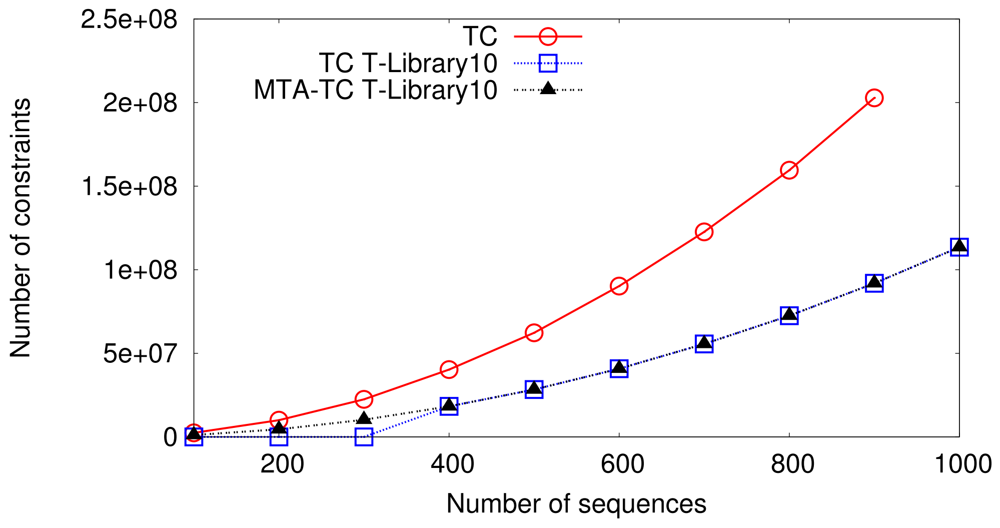


Figure 5.9: MTA-TC T-Library execution times increasing the number of cores

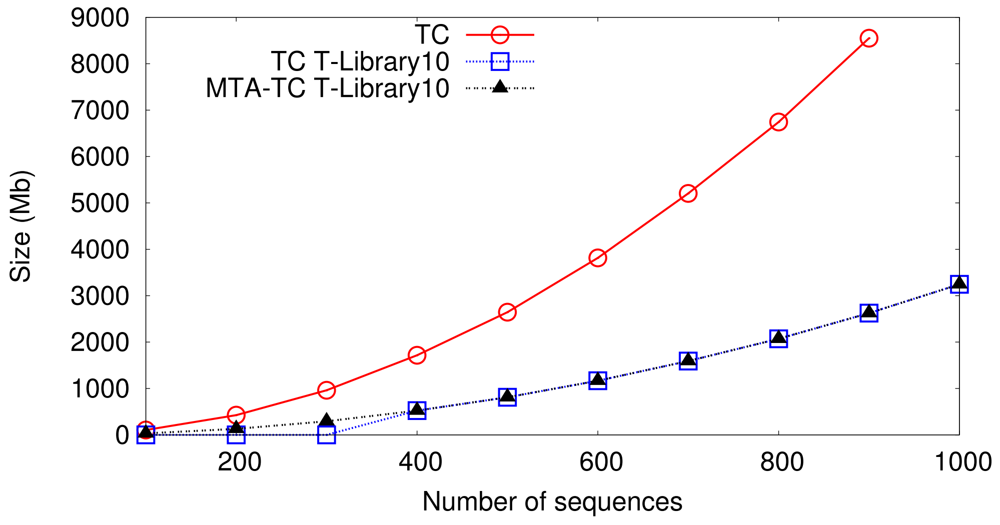
It can be seen that the scalability behavior of both MTA-TC T-Library10 and MTA-TC T-Library50 was similar to MTA-TC and the execution time fell when the number of cores was increased. It can also be seen that, in the worst case (8 cores), the consistency optimization allowed the execution time of MTA-TC to be reduced by 47.09% for MTA-TC T-Library10 and 80.06% for MTA-TC T-Library50. However, in the ideal mapping case (96 cores), the reduction in execution time was 39.15% for MTA-TC T-Library10 and 49.24% for MTA-TC T-Library50. In other words, MTA-TC T-Library10 and MTA-TC T-Library50 were respectively 47.93% and 37.58% slower than TC, while MTA-TC was 68% slower. As explained in Section 5.5.3, this time penalty in TC is due to each worker has to read the library created by the master.

The second experiment is the same as the one presented in Section 4.4.2, but in this case we compared the scalability of parallel MTA-TC T-Library10 with TC T-Library10 and standard TC (Figures 5.10a, 5.10b, 5.10b). The experiment was run on an 8-node cluster where each node comprised a 2.1GHz AMD Opteron and 12.5GB of RAM. Thus, the experiments consisted of running first TC and TC T-Library10 on a node in the cluster, and then, launching MTA-TC T-Library10 in parallel building 7 guide trees, which was the ideal number of tasks because the cluster was made up of only 8 cores: a master and 7 slaves. Sequences sets with varying the number of sequences were used as input data.

Figures 5.10a and 5.10b show the comparison of the number of constraints in the library and its size in Mb respectively, with the number of sequences increasing from 100 to 900.



(a) Analysis of the number of constraints



(b) Analysis of the memory requirements

Figure 5.10: Study of the MTA-TC T-library memory requirements

As expected, the memory size and number of constraints was the same in the TC T-Library10 and MTA-TC T-Library10 because both approaches use the same optimization configuration. It can be seen that the MTA-TC T-Library10 significantly reduced the number of constraints and the library size compared with TC. With 900 sequences, the MTA-TC T-Library10 reduced the number of constraints and the size of the TC default library by 69.31%. This study also demonstrates that the MTA-TC T-Library10 is more scalable than TC, because TC cannot align 1000 sequences while MT-TC T-library10 is able to align them. Moreover, as demonstrated in Section 4.4.2 and as seen by the memory consumption in Figure 5.10, MTA-TC T-Library10 could align up to twice as many sequences as can be aligned with TC.

Otherwise, Figure 5.11 shows the comparison of the total execution time between the MTA-TC T-Library10, TC T-Library10 and TC. It can be seen that the MTA-TC T-Library10 is faster than the standard TC, but it is slightly slower than the TC T-Library10. Specifically, with 900 sequences, MTA-TC using T-Library10 reduced the execution time of default TC by 76.05%. However, it was 11.6% slower than TC using the T-Library10 approach, due to the same problem explained above that all workers have to read the library from the master.

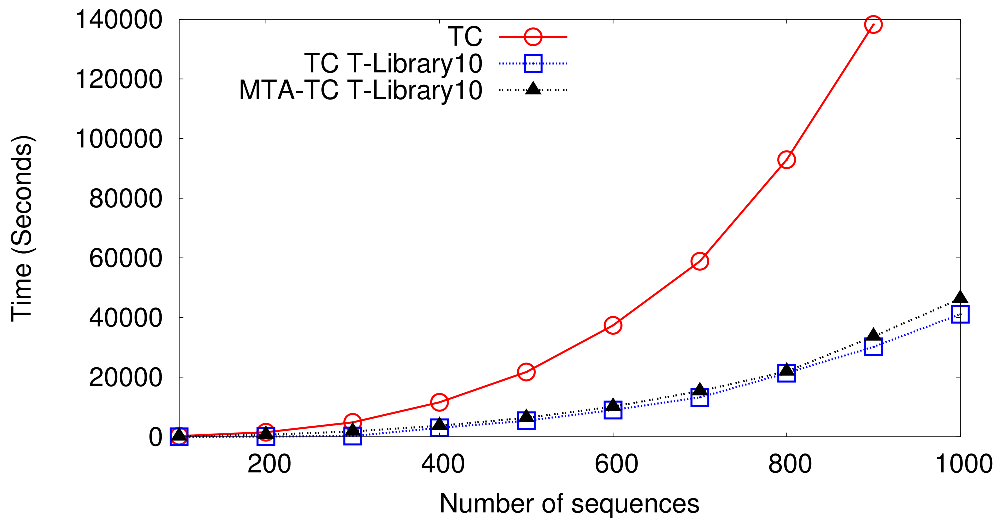


Figure 5.11: Analysis of the MTA-TC T-library total execution times

5.6.3.2 Sequential Multiples Tree Alignment

After analyzing the parallel version of MTA, given the integration of the consistency library optimization proposal with MTA-TC, it is necessary to study the performance of the sequential version of MTA-TC and compare it to the standard TC. Accordingly, this experiment presents the comparison of accuracy and execution time between the standard TC and the serial version of MTA-TC aligning 5 trees. The reason for choosing 5 trees is that, as seen in experimental tests, it is the maximum number of trees in that the execution time of MTA-TC T-Library10 is below the TC runtime. Therefore, this experiment seeks to demonstrate that, using the consistency optimization technique from TC and aligning 5 trees, the sequential version of MTA is more accurate and even less time expensive than the original TC method.

Table 5.6 presents the experimental results. For the accuracy results, the average quality of the alignments of TC, MTA-TC T-Library10 and MTA-TC T-Library50, were compared using NiRMSD as a selection metric. The experiment was done using the whole PREFAB benchmark sequence sets, divided into five groups according to the percent identity of the sequences, and the Q score to measure the accuracy. For the runtime results, we compared the execution time of these three approaches aligning a sequence set formed by 900 sequences. In the table, the first four columns identify the range of identity and the last column indicates the total execution time.

The results show that, aligning only 5 trees, MTA-TC T-Library10 improved the alignment accuracy of TC by 0.4% (0.712 vs. 0.709). However T-MTA-TC T-Library50, aligning 5 trees, was not able to obtain the same accuracy results as TC due to the reduction of consistency information, and it was 0.84% less accurate (0.703 v. 0.709). Regarding the execution time re-

Aligner	**0 - 15**	**15 - 25**	**25 - 35**	**35 - 100**	**0 - 100**	**Time (s)**
T-Coffee	0.421	0.721	0.876	0.951	0.709	138251.223
MTA-TC T-Library10	0.429	0.722	0.876	0.955	0.712	130231.630
MTA-TC T-Library50	0.409	0.713	0.874	0.957	0,703	80695.930

Table 5.7: Accuracy and execution time analysis of the sequential MTA-TC T-library approaches

sults, Table 5.7 indicates that due to the memory optimization approach, even aligning 5 trees in a sequential way, MTA-TC T-Library10 was 5.8% faster than TC, while MTA-TC T-Library50, which implements a more aggressive reduction of memory approach, was 41.63% faster. Therefore, there is still capacity to increase the number of trees analyzed by MTA-TC T-Library50 in order to improve its accuracy, without exceeding the execution time of TC.

Moreover, it is important to note that MTA-TC T-Library is more scalable in both dimensions: the number of sequences to be aligned and the accuracy of the final alignment. The factor of memory reduction can be increased to align more sequences and to enable more guide trees to be generated in order to improve the alignment.

In summary, this experiment demonstrates that the integration of the consistency optimization approach in TC with the sequential version of MTA method allows the alignment accuracy of TC to be increased without penalizing the execution time due to the completion of more alignments in a serial way.

5.7 Multiple Tree Alignment Conclusions

In summary, this chapter presents MTA, a new MSA method designed to prevent the problem of progressive alignment strategies caused by errors produced by the guide tree. The proposed methodology consists of building multiple guide trees from the same input sequences, aligning them with an existing MSA program and finally evaluating the resulting alignments to select the best one as the final result.

Although MTA accepts any progressive alignment program that allows guide trees as the input parameter, the aligners used during the present work were ClustalW (MTA-CLW) and T-Coffee (MTA-TC). Six single scores were used to evaluate the resulting alignments. In addition, two heuristic meta-score metrics (WSC and MCC) that were obtained by using genetic algorithms were used to find a good combination between the previous six single metrics. While WSC is a weighted scheme, where each input metric is assigned a weight to maximize the correlation with the biological quality, MCC is a more complex scheme that calculates the meta-score through a program code that takes the relationships between the input metrics into account.

The scoring metrics study showed that the two meta-scores are the best metrics for evaluating and selecting the appropriate alignment, followed by the metrics that use structural information. The experimental results also showed that using the best configurations of MTA-CLW and MTA-TC improved the alignment accuracy by 3.10% and 15.40% over the original MSA methods respectively. The scalability study demonstrated that the parallel version of MTA is scalable with an increasing number of processors. Despite analysing more guide trees, MTA is capable of producing the alignments in a similar time to the original methods.

Regarding the integration of MTA with TC using the consistency library optimization approach presented in Chapter 4, we demonstrated that the use of MTA can recover the accuracy loss caused by the reduction in consistency information. On the other hand, the use of TC with the memory optimization approach allows a reduction in the execution time needed to align N trees in MTA-TC. Furthermore, this integration also allows MTA to align more trees in a serial way in order to improve the quality of TC, without increasing its execution time and improving the scalability of T-Coffee.

Finally, although the accuracy results of our proposal are close to the maximum ones shown in Figure 5.2, none of the implemented scores is capable of reaching the accuracy values obtained using a benchmark score as the selection metric. This is because benchmarks use more information to evaluate alignments, mainly their reference alignments. A major challenge in biology is to find an evaluation score without reference able to choose the most accurate of several alignments and obtain accuracy results similar to the maximum accuracy values obtained in Figure 5.2.

Chapter 6

Conclusions and Future Work

This chapter describes the conclusions reached in this work, together with the main contributions and publications. Moreover, it also explains the main open lines that should be developed in the near future.

6.1 Conclusions

Due to the entry into the area of comparative genomics, the simultaneous comparison of a large number of homologous sequences has become increasingly important. Therefore, there is no doubt that Sequence Alignment, in particular Multiple Sequence Alignment (MSA), is by far the most common task in bioinformatics. MSA constitutes an extremely powerful means of revealing the constraints imposed by structure and function on the evolution of a protein family. However, MSA is a complex problem, whose solution stands at a crossroads between biology and computation.

The biological issue surrounding MSAs lies in the definition of correctness. Most MSA methods define an objective function that defines the mathematical objective of the search. The main problem is that the perfect objective function that defines the mathematically optimal alignment not guaranteed to be biological optimal. Another important biological issue is that defining the proper objective function is a highly non-trivial task and an important research area.

Regarding the computational issue, the computation of a mathematically optimal alignment is a NP-Complete problem. For this reason, all the current implementations of MSA algorithms are heuristics and none of them guarantees a full optimization.

Progressive alignment is by far the most widely used heuristic. Progressive methods assemble a multiple alignment by making a series of pairwise alignments of sequences where sequences or alignments are added one by one, depending on the order established by a guide tree. Although this heuristic provides a great advantage of speed and simplicity, progressive methods are very dependent on the initial alignments, and several studies have shown that the alignment may be sensitive to errors in the guide tree. To correct or minimize errors made in progressive alignment steps, two techniques are frequently used: iterative refinement and consistency scoring. Iterative refinement is based on carrying out a progressive alignment and then refining the result by repeatedly dividing the aligned sequences into sub-alignments and realigning the sub-alignments. On the other hand, consistency-based methods use sequence information to avoid mistakes in the alignment. However, the introduction of more information leads to increased memory requirements.

The new challenges in genomics, the exponential growth of biological data and the inability to treat it efficiently have led to many of the existing methods becoming obsolete due to them not being capable of aligning thousands or even hundreds of thousands of sequences. This problem has highlighted the need for an interrelationship between biologists, bioinformatics and computer scientists. Some MSA methods introduced High Performance Computing capabilities to take advantage of the new technologies and infrastructures. However, all of these have exhibited scalability problems when the number of sequences increases, as they are constrained by data dependencies that guide the alignment process.

Through collaboration with a group at the Centre for Genomic Regulation (CRG), we have worked with T-Coffee and its parallel version Parallel-T-Coffee. T-Coffee is one of the most popular MSA methods that combines a consistency-based scoring function with the progressive alignment algorithm. Although the consistency-scheme reduces the errors caused by the guide tree so obtaining more accurate alignments, it is known that the introduction of consistency information increases the requirements for CPU time and memory thus reducing the scalability and limiting the number of sequences the method is able to align.

This thesis presents three proposals for minimizing the three problems presented above: The scalability problems of MSA parallel implementations because of data dependencies, the limited scalability of consistency-based methods due to huge memory requirements, and finally, the biological accuracy problems caused by the high dependency of the guide tree order.

The first presented proposal, called Balanced Guide Tree (BGT), is devoted to solving the scalability problems of MSA parallel implementations while maintaining the accuracy of the alignments. BGT is a new guide tree construction heuristic, based on the neighbor-joining clustering algorithm, that consists of modifying the tree generation method to take into account not only the similarity between sequences, but also balancing features. BGT is designed to produce more balanced guide trees in order to eliminate the bottleneck generated by the high dependencies between different iterations of the progressive alignment step. Balancing studies demonstrated that our proposed method generated more balanced guide trees than the standard one, resulting in a significant reduction in the critical path and an important rise in the number of tasks that can be executed in parallel.

The BGT is not only able to improve the performance of T-Coffee but also does so without losing quality in the resulting alignment. In the majority of the datasets analyzed, BGT obtained better accuracy.

Although BGT is designed to be used or implemented in any parallel progressive alignment or iterative method that works with neighbor-joining guide trees, in this thesis, BGT was implemented and evaluated in concurrent T-Coffee (Balanced-TCoffee) and its parallel distributed-memory version Parallel-TCoffee (BalancedParallel-TCoffee). We showed that BalancedParallel-TCoffee can take advantage of large high-performance computing infrastructures to reduce the execution time of MSA applications. More specifically, the experimental results showed that on a 100-node cluster, BalancedParallel-TCoffee reduced the execution time of Parallel-TCoffee by 68%. Regarding Balanced-TCoffee, the scalability study showed that it was also able to improve the performance of multi-process T-Coffee and exploit the computing resources of a multi-core workstation. In particular, Balanced-TCoffee, which was run on a 24 cores workstation, improved the execution time of T-Coffee by 12%. Finally, the experimental results indicated that run-time performance is achieved while maintaining the biological accuracy of the resulting alignments.

This contribution led to the following publications:

[OCGM09] M. Orobitg, F. Cores and F. Guirado. *Exploiting Performance on Parallel T-Coffee Progressive Alignment by Balanced Guide Tree.* Proceedings of the CMMSE 2009, vol. 3, pp. 793-805, 2009.

[OGNC09] M. Orobitg, F. Guidado, F. Cores and C. Notredame. *Exploiting Parallelism on Progressive Alignment Methods.* Journal of Supercomputing, vol.58(2), pp. 186–194, 2009.

The second proposal presented focused on the huge memory and runtime requirements presented of MSA consistency-based methods. The proposed method consists of an optimization method for the T-Coffee library to reduce the memory and CPU time requirements.

Two approaches to optimization were defined: The first one identifies the information that will be useful during the alignment stage and the information that can be discarded without affecting the quality of the alignment excessively. The second one discards all the residues that are below a threshold defined by the user. This second approach provides the user with greater flexibility to choose how aggressive the reduction of the library can be in order to trade off between alignment time and quality. Furthermore, the library optimization approaches were developed in the kernel of T-Coffee alignment tool. This means that not only is the default T-Coffee tool able to benefit from these improvements but also the performance and scalability of all package tools, like M-Coffe, R-Coffee and Expresso, can be improved.

The proposed solution was implemented in concurrent T-Coffee and evaluated by comparing T-Coffee using our optimized library with the standard T-Coffee. The scalability results showed that the optimized library is able to decrease the execution time, enhance the scalability and performance of the application considerably, and thus increase the number of sequences that can be aligned. For instance, the results of one test in the experimentation showed that our optimization approach decreases the memory requirements of T-Coffee by 75%, reduces the execution time by 92%, and finally, allows T-Coffee to align 2000 sequences while the standard T-Coffee is only able to align 1000 sequences. Regarding the accuracy results, the quality of the alignments obtained by T-Coffee using the optimization library algorithm is worse than the original obtained with the standard T-Coffee because the consistency information is reduced. However, this loss in quality is limited to less than 3%. Even with this, they are more accurate than other library optimization methods and other MSA methods of the literature. Finally, the improvement presented can widen the range of scenarios in which T-Coffee can be used efficiently as an alignment tool.

This proposed method and its results were presented in the following publications:

[OCG+12b] M. Orobitg, F. Cores, F. Guirado, C. Kemena, C. Notredame and A. Ripoll. *Enhancing the scalability of consistency-based progressive multiplesequences alignment applications.* Proceedings of the 26th International Parallel and Distributed Processing Symposium (IPDPS), 2012.

Finally we proposed a new MSA method, called Multiple Tree Alignment (MTA), designed to cope with the errors from the progressive alignment strategies caused by the guide tree in order to improve the biological accuracy. The proposed methodology consisted of building multiple guide trees from the same input sequences, aligning them with an existing MSA program and finally evaluating the resulting alignments to select the best one as the final result.

Although MTA accepts any progressive alignment program that accepts guide trees as input parameter, MTA was evaluated using ClustalW (MTA-CLW) and T-Coffee (MTA-TC). The first one is faster for aligning larger alignments, while the second one is slower but more accurate.

To evaluate the resulting alignments, MTA used six single scores found in the literature: two metrics that incorporate structural information to evaluate the alignment, and four that do not. However, in this thesis, two new heuristic meta-score are proposed (WSC and MCC). These meta-scores were obtained by using genetic algorithms to find a good combination between the previous six single metrics. While WSC is a weighted scheme, where each input metric is assigned a weight to maximize the correlation with the biological quality, MCC is a more complex scheme that calculates the meta-score through a program code that takes the relationships between the input metrics into account.

The study of the scoring metrics showed that both proposed meta-scores are the best metrics for evaluating and selecting the appropriate alignment, followed by metrics that use structural information. The accuracy results also showed that our proposed methods not only are more accurate than T-Coffee and ClustalW, but also more accurate than other MSA methods. For example, using the best configurations of MTA-CLW and MTA-TC, the alignment accuracy improved by 3.10% and 15.40% over ClustalW and T-Coffee respectively. The scalability study demonstrated that the parallel version of MTA is scalable with an increasing number of processors. Despite analyzing more guide trees, MTA is capable of producing the alignments in a similar time to the original methods.

The following two articles were published from the meta-scores work:

[OCG12a] M. Orobitg, F. Cores and F. Guirado. *MSA score accuracy analysis based on genetic algorithms.* Proceedings of the CMMSE 2012, pp. 935-946, 2012.

[OCG^{+}13] M. Orobitg, F. Cores, F. Guirado, C. Roig and C. Notredame. *Improving Multiple Sequence Alignment biological accuracy through genetic algorithms.* Journal of Supercomputing, 2013.

The results from the MTA contribution are pending submission to the following journal and are being reviewed:

Pending submission M. Orobitg, F. Guirado, J.Ll. Lerida, J. Lladós, C. Notredame and F. Cores. *MTA: A MSA method to increase alignment accuracy by evaluating multiple guide trees.* Bioinformatics.

The last experiment consisted of integrating the last two proposals: MTA with T-Coffee using the consistency library optimization approach. The results demonstrated that the use of MTA can not only recover the accuracy loss caused by the reduction in the consistency information, but also be more accurate than T-Coffee. On the other hand, the scalability study demonstrated that the optimized memory MTA-TC is also a scalable method, and the use of T-Coffee with the memory optimization approach allows parallel MTA-TC to reduce the execution time of the standard MTA-TC. Furthermore, this runtime analysis also indicated that the memory optimized MTA-TC is able to align more trees serially to improve the quality of TC, without increasing its execution time and maintaining the scalability.

Finally, through the collaboration and a research stay with the Centre for Genomic Regulation group, two more publications have been developed:

[TOG+10] P.D. Tommaso, M. Orobitg, F. Guirado, F. Cores, T. Espinosa and C. Notredame. *Cloud-Coffee: Implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud.* Bioinformatics, col.26(15), pp. 1903-1904, 2010.

[TMX+11] P.D. Tommaso, S. Moretti, I. Xenarios, M. Orobitg, A. Montanyola, J.M. Chang, J.F. Taly and C. Notredame. *T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension.* Nucleic acids research, Web Server issue, vol. 39, W13–W17, 2011.

6.2 Future Work

At this point, we can say that this thesis has covered all its objectives. From a research perspective, based on the extensive knowledge during its development, this thesis has also opened some new challenges to be tackled.

1. **New pairwise profile alignments.** This issue can be considered as one of the most important weaknesses of our proposed heuristic Balanced Guide Tree. Existing pairwise profile alignments are not designed

to treat balanced guide trees. It is more time expensive for this algorithm to align two profiles with a similar number of sequences than to align two unbalanced profiles. As is known, during the final progressive alignment steps and due to the use of balanced trees, the method tends to obtain large balanced profiles. In conclusion, these algorithms sometimes penalize the improvements obtained by increasing the degree of parallelism of the balanced trees. Accordingly, new pairwise profile alignments are needed to take advantage of the higher degree of parallelism of balanced guide trees and thus improve the scalability of this progressive alignment process.

2. **New parallel algorithms for MSA.** It is well known that the MSA method has to be adapted to the new data-intensive era. For this reason, not only do MSA methods have to be redesigned to take advantage of increasing resources, but also new parallel techniques need to be designed to take advantage of new parallel paradigms (mapreduce, CUDA), new architectures (Hadoop, Cloud, P2P) or special hardware (GPUs).

3. **New distributed consistency-based scheme.** One of the main drawbacks of consistency-based MSA methods is their huge memory requirements. These grow exponentially when the number of sequences and their length increase. Nowadays, these methods have been discarded for aligning large sequence sets and their future is limited to aligning few sequences. An interesting work could be to redesign and implement new techniques for adapting these consistency structures to distributed memory environments, such as distributed databases.

4. **New evaluation meta-scores.** New improvements to the configuration of genetic algorithm can be studied to improve the quality of the evaluation function. The introduction of new scores, new conditions for the evolution or improving the training data are possible new research lines for developing new and more sensitive scores.

5. **Large-Scale aligners.** With the increasing performance of sequencing hardware, it is not so far the necessity of aligning the genome of hundred

or thousands of individuals. Taking the huge volume of information required for this task into consideration, Large-Scale aligners will require a drastic change in design. Some consistency may be mandatory to guarantee a minimal quality for those alignments. However, to guarantee scalability, their memory requirements have to be limited. The intensive use of HPC infrastructures is required in order to address this problem. Moreover, it is essential to facilitate the use of such infrastructures by biologists and bioinformatics. Cloud platforms will carry out a leading role on achieving this goal.

Bibliography

[AEAT$^+$07] M. Abouellail, E. El-Araby, M. Taker, T. El-Ghazawi, and G.B. Newby. Dna and protein sequence alignment with high performance reconfigurable systems. In *Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on*, pages 334 –341, aug. 2007.

[AFG$^+$09] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. *Above the Clouds: A Berkeley View of Cloud Computing.* EECS Department, University of California, Berkeley, Feb 2009.

[AGM$^+$90] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990.

[AL89] S. Altschul and D.J. Lipman. Trees, Stars, and Multiple Biological Sequence Alignment. *SIAM Journal on Applied Mathematics*, 49(1):197–209, 1989.

[Alt91] S. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219(3):555–565, June 1991.

[Amd67] G.M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM.

[AMKN06] F. Armougom, S. Moretti, V. Keduas, and C. Notredame. The irmsd: a local measure of sequence alignment accuracy using structural information. In *ISMB (Supplement of Bioinformatics)*, pages 35–39, 2006.

[BKML$^+$11] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, and E.W. Sayers. GenBank. *Nucleic acids research*, 39(Database issue):D32–D37, January 2011.

[BKR05] M. Bauer, G.W. Klau, and K. Reinert. Multiple structural rna alignment with lagrangian relaxation. In *Proceedings of the 5th International conference on Algorithms in Bioinformatics*, WABI'05, pages 303–314, Berlin, Heidelberg, 2005. Springer-Verlag.

[BLB09] K. Benkrid, Y. Liu, and A. Benkrid. A highly parameterized and efficient fpga-based skeleton for pairwise biological sequence alignment. *IEEE Trans. Very Large Scale Integr. Syst.*, 17(4):561–570, April 2009.

[BM06] R.B. Batista and A.C. Magalhaes. Z-align: An exact and parallel strategy for local biological sequence alignment in user-restricted memory space. In *Proceedings of the 2006 IEEE International Conference on Cluster Computing, September 25-28, 2006, Barcelona, Spain.* IEEE, 2006.

[BMP$^+$03] M. Brudno, S. Malde, A. Poliakov, C.B. Do, O. Couronne, I. Dubchak, and S. Batzoglou. Glocal alignment: finding rearrangements during alignment. *Bioinformatics*, 19(suppl 1):i54–i62, July 2003.

[BP66] L.E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The Annals of Mathematical Statistics*, 37(6):1554–1563, 1966.

[BRS$^+$09] R.K. Bradley, A. Roberts, M. Smoot, S. Juvekar, J. Do, C.N. Dewey, I. Holmes, and L. Pachter. Fast statistical alignment. *PLoS Computational Biology*, 5(5), 2009.

[BS87] C.J. Barton and M.J.E. Sternberg. Evaluation and improvements in the automatic alignment of protein sequences. *Protein Eng.*, 1(2):89–94, 1987.

[BWF$^+$00] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[Che09] S.J. Chen. *Hardware Software Co-design of a Multimedia Soc Platform*. Springer, 2009.

[CKT06] K. Chaichoompu, S. Kittitornkun, and S. Tongsima. Mt-clustalw: multithreading multiple sequence alignment. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 8 pp., april 2006.

[CL88] H. Carrillo and D.J. Lipman. The Multiple Sequence Alignment Problem in Biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

[CN06] J.M. Claverie and C. Notredame. *Bioinformatics For Dummies*. For Dummies, 2 edition, December 2006.

[Cor88] F. Corpet. Multiple sequence alignment with hierarchical clustering. *Nucl. Acids Res.*, 16(22):10881–10890, November 1988.

[Cri58] F.H.C. Crick. On Protein Synthesis. *The Symposia of the Society for Experimental Biology*, 12:138–163, 1958.

[Cri70] F.H.C. Crick. Central Dogma of Molecular Biology. *Nature*, 227(5258):561–563, August 1970.

[CS03] C. Chen and B. Schmidt. Computing large-scale alignments on a multi-cluster. In *CLUSTER*, pages 38–45, 2003.

[DEKM98] R. Durbin, S.R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, July 1998.

[Den03] E.V. Denardo. *Dynamic Programming: Models and Applications*. Dover Publications, Mineola, New York, dover books on computer science edition, 2003.

[DKC05] J. Dollimore, T. Kindberg, and G. Coulouris. *Distributed Systems: Concepts and Design (4th Edition) (International Computer Science Series)*. Addison Wesley, May 2005.

[DLS$^+$06] A. Driga, P. Lu, J. Schaeffer, D. Szafron, K. Charter, and I. Parsons. Fastlsa: A fast, linear-space, parallel and sequential algorithm for sequence alignment. *Algorithmica*, 45(3):337–375, July 2006.

[DMBB05] C.B. Do, M.S. Mahabhashyam, M. Brudno, and S. Batzoglou. ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome research*, 15(2):330–340, February 2005.

[DS78] M.O. Dayhoff and R.M. Schwartz. Chapter 22: A model of evolutionary change in proteins. In *in Atlas of Protein Sequence and Structure*, 1978.

[EB06] R.C. Edgar and S.L. Batzoglou. Multiple sequence alignment. *Current opinion in structural biology*, 16(3):368–373, 2006.

[Edd95] S.R. Eddy. Multiple alignment using hidden markov models. *Proc Int Conf Intell Syst Mol Biol*, 3:114–120, 1995.

[Edg04] R.C. Edgar. Local homology recognition and distance measures in linear time using compressed amino acid alphabets. *Nucleic Acids Res*, 32(1):380–385, 2004.

[FD87] D.F. Feng and R.F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of molecular evolution*, 25(4):351–360, 1987.

[Fly72] M.J. Flynn. Some Computer Organizations and Their Effectiveness. *IEEE Transactions on Computers*, C-21(9):948–960, September 1972.

[Fos95] I. Foster. *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering.* Parallel programming / scientific computing. Addison-Wesley, 1995.

[Fos05] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. In *NPC*, volume 3779 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2005.

[GL04] C. Grasso and C. Lee. Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems. *Bioinformatics*, 20(10):1546–1556, July 2004.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[Got90] O. Gotoh. Consistency of optimal sequence alignments. *Bull Math Biol*, 52(4):509–525, 1990.

[Got96] O. Gotoh. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. *Journal of molecular biology*, 264(4):823–838, December 1996.

[HH84] P. Hogeweg and B. Hesper. The alignment of sets of sequences and the construction of phyletic trees: An integrated method. *Journal of Molecular Evolution*, 20(2):175–186, June 1984.

[HH92] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences of the United States of America*, 89(22):10915–10919, November 1992.

[HJS00] M.D. Hill, N.P. Jouppi, and G. Sohi. *Readings in Computer Architecture*. The Morgan Kaufmann Series in Computer Architecture and Design Series. Morgan Kaufmann, 2000.

[HK96] R. Hughey and A. Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Computer applications in the biosciences : CABIOS*, 12(2):95–107, April 1996.

[HM91] X. Huang and W. Miller. A time-efficient, linear-space local similarity algorithm. *Advances in Applied Mathematics*, 12:337–357, 1991.

[Hol92] J.H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. A Bradford Book, April 1992.

[HS88] D.G. Higgins and P.M. Sharp. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, December 1988.

[HT89] A.R. Hoffman and J.E. Traub. *Supercomputers: directions in technology and applications*. National Academy Press, 1989.

[HX98] K. Hwang and Z. Xu. *Scalable parallel computing: technology, architecture, programming*. Computer engineering series. WCB/McGraw-Hill, 1998.

[IEE91] IEE. *IEEE standard computer dictionary : a compilation of IEEE standard computer glossaries*. IEEE Computer Society Press, New York, NY, USA, January 1991.

[JAA06] B. Javadi, M.K. Akbari, and J.H. Abawajy. A performance model for analysis of heterogeneous multi-cluster systems. *Parallel Comput.*, 32(11-12):831–851, December 2006.

[JMF09] S. Jha, A. Merzky, and G. Fox. Using clouds to provide grids with higher levels of abstraction and explicit support for usage modes. *Concurr. Comput. : Pract. Exper.*, 21:1087–1108, June 2009.

[Jun09] S. Jung. Parallelized pairwise sequence alignment using cuda on multiple gpus. *BMC Bioinformatics*, 10(S-7), 2009.

[Jus01] W. Just. Computational complexity of multiple sequence alignment with SP-score. *Journal of computational biology: a journal of computational molecular cell biology*, 8(6):615–638, 2001.

[KA99] Y. Kwok and I. Ahmad. Benchmarking and comparison of the task graph scheduling algorithms. *J. Parallel Distrib. Comput.*, 59(3):381–422, 1999.

[Kab76] W. Kabsch. A Solution for the Best Rotation to Relate Two Sets of Vectors. *Acta Crystallographica*, 32:922–923, 1976.

[Kab78] W. Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, September 1978.

[KBM$^+$94] 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, February 1994.

[Kim80] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16(2):111–120, December 1980.

[KMKM02] K. Katoh, K. Misawa, K. Kuma, and T. Miyata. MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, 30(14):3059–3066, July 2002.

[KN09] C. Kemena and C. Notredame. Upcoming challenges for multiple sequence alignment methods in the high-throughput era. *Bioinformatics (Oxford, England)*, 25(19):2455–2465, October 2009.

[KPC94] J. Kim, S. Pramanik, and M.J. Chung. Multiple sequence alignment using simulated annealing. *CABIOS*, 10(4):419–426, July 1994.

[KTKN11] C. Kemena, J.F. Taly, J. Kleinjung, and C. Notredame. Strike: evaluation of protein msas using a single 3d structure. *Bioinformatics*, 27(24):3385–3391, 2011.

[KW85] C.P. Kruskal and A. Weiss. Allocating independent subtasks on parallel processors. *IEEE Trans. Software Eng.*, 11(10):1001–1016, 1985.

[LAK89] D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences*, 86(12):4412–4415, June 1989.

[Li03] K. Li. Clustalw-mpi: Clustalw analysis using distributed and parallel computing. *Bioinformatics*, 19(12):1585–1586, 2003.

[LR96] T. Lengauer and M. Rarey. Computational methods for biomolecular docking. *Curr Opin Struct Biol*, 6(3):402–406, June 1996.

[LS05] T. Lassmann and E. Sonnhammer. Kalign - an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, 6(1):298+, 2005.

[LSM09] Y. Liu, B. Schmidt, and D.L. Maskell. Msa-cuda: Multiple sequence alignment on graphics processing units with cuda. In *ASAP*, pages 121–128. IEEE, 2009.

[LSM10] Y. Liu, B. Schmidt, and D.L. Maskell. Msaprobs: multiple sequence alignment based on pair hidden markov models and partition function posterior probabilities. *Bioinformatics*, 26(16):1958–1964, 2010.

[LSVMW06] B. Liu, B. Schmidt, G. Voss, and W. Müller-Wittig. Gpu-clustalw: Using graphics hardware to accelerate multiple sequence alignment. In Yves Robert, Manish Parashar, Ramamurthy Badrinath, and Viktor K. Prasanna, editors, *HiPC*, volume 4297 of *Lecture Notes in Computer Science*, pages 363–374. Springer, 2006.

[Mar71] A. Markov. Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In R. Howard, editor, *Dynamic Probabilistic Systems (Volume I: Markov Models)*, chapter Appendix B, pages 552–577. John Wiley & Sons, Inc., New York City, 1971.

[MDBO98] K. Mizuguchi, C.M. Deane, T.L. Blundell, and J.P. Overington. HOMSTRAD: a database of protein structure alignments for homologous families. *Protein science : a publication of the Protein Society*, 7(11):2469–2471, November 1998.

[MFDW98] B. Morgenstern, K. Frech, A. Dress, and T. Werner. DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14(3):290–294, April 1998.

[MM88] E.W. Myers and W. Miller. Optimal alignments in linear space. *Computer applications in the biosciences : CABIOS*, 4(1):11–17, March 1988.

[Mou04] D.W. Mount. *Bioinformatics: Sequence and Genome Analysis, Second Edition.* Cold Spring Harbor Laboratory Press, 2nd edition, July 2004.

[Mou07] D.W. Mount. Dot Matrix Pairwise Sequence Comparison. *Cold Spring Harbor Protocols*, 2007(24), 2007.

[MRR$^+$53] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[MVF94] M.A. McClure, T.K. Vasi, and W.M. Fitch. Comparative analysis of multiple protein-sequence alignment methods. *Mol. Biol. Evol.*, 11(4):571–592, 1994.

[NH96] C. Notredame and D.G. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic acids research*, 24(8):1515–1524, April 1996.

[NHH98] C. Notredame, L. Holm, and D.G. Higgins. COFFEE: an objective function for multiple sequence alignments. *Bioinformatics*, 14(5):407–422, June 1998.

[NHH00] C. Notredame, D.G. Higgins, and J. Heringa. T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, September 2000.

[NMA$^+$10] T. Nilsson, M. Mann, R. Aebersold, J.R. Yates, A. Bairoch, and J.J Bergeron. Mass spectrometry in high-throughput proteomics: ready for the big time. *Nature methods*, 7(9):681–685, September 2010.

[NOH97] C. Notredame, E.A. O'Brien, and D.G. Higgins. RAGA: RNA sequence alignment by genetic algorithm. *Nucleic acids research*, 25(22):4570–4580, November 1997.

[Not02] C. Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, January 2002.

[NW70] 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(3):443–453, March 1970.

[OCG12a] M. Orobitg, F. Cores, and F. Guirado. Msa score accuracy analysis based on genetic algorithms. In *CMMSE'12: 12th International Conference Computational and Mathematical Methods in Science and Engineering*, pages 935–946, 2012.

[OCG$^+$12b] M. Orobitg, F. Cores, F. Guirado, C. Kemena, C. Notredame, and A. Ripoll. Enhancing the scalability of consistency-based progressive multiple sequences alignment applications. In *IPDPS'12: 26th International Parallel and Distributed Processing Symposium*, 2012.

[OCG$^+$13] M. Orobitg, F. Cores, F. Guirado, C. Roig, and C. Notredame. Improving multiple sequence alignment biological accuracy through genetic algorithms. *The Journal of Supercomputing*, pages 1–13, 2013.

[OCGM09] M. Orobitg, F. Cores, F. Guirado, and A. Montañola. Exploiting performance on parallel t-coffee progressive alignment by balanced guide tree. In *CMMSE '09: 9th International Conference Computational and Mathematical Methods in Science and Engineering*, pages 793–805, 2009.

[OGNC09] M. Orobitg, F. Guirado, C. Notredame, and F. Cores. Exploiting parallelism on progressive alignment methods. *The Journal of Supercomputing*, 58(2):186–194, 2009.

[OHL$^+$08] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. Gpu computing. *Proceedings of the IEEE*, 96(5):879 –899, may 2008.

[OSA$^+$04] O. O'Sullivan, K. Suhre, C. Abergel, D.G. Higgins, and C. Notredame. 3D-Coffee: combining protein sequences and structures within multiple sequence alignments. *Journal of molecular biology*, 340(2):385–395, July 2004.

[OZH$^+$03] O. O'Sullivan, M. Zehnder, D.G. Higgins, P. Bucher, A. Grosdidier, and C. Notredame. Apdb: a novel measure for benchmarking sequence alignment methods without reference alignments. In *ISMB (Supplement of Bioinformatics)*, pages 215–221, 2003.

[PG07] J. Pei and N.V. Grishin. PROMALS: towards accurate multiple sequence alignments of distantly related proteins. *Bioinformatics*, 23(7):802–808, April 2007.

[PK87] C.D. Polychronopoulos and D.J. Kuck. Guided self-scheduling: A practical scheduling scheme for parallel supercomputers. *IEEE Trans. Computers*, 36(12):1425–1439, 1987.

[PL88] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences*, 85(8):2444–2448, April 1988.

[PPL$^+$10] O. Penn, E. Privman, G. Landan, D. Graur, and T. Pupko. An alignment confidence score capturing robustness to guide tree uncertainty. *Mol Biol Evol*, 27(8):1759–67, 2010.

[RA04] S. Rajko and S. Aluru. Space and time optimal parallel sequence alignments. *IEEE Trans. Parallel Distrib. Syst.*, 15(12):1070–1081, December 2004.

[Rob04] E. Robert. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, 5(1):113+, August 2004.

[RSA$^+$03] G.P.S. Raghava, S. Searle, P. Audley, J. Barber, and G. Barton. OXBench: A benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC Bioinformatics*, 4(1):47+, October 2003.

[RULD06] Roshan, Usman, Livesay, and R. Dennis. Probalign: multiple sequence alignment using partition function posterior probabilities. *Bioinformatics*, 22(22):2715–2721, November 2006.

[SB05] S. Siebert and R. Backofen. MARNA: multiple alignment and consensus structure prediction of RNAs based on sequence structure comparisons. *Bioinformatics*, 21(16):3352–3359, August 2005.

[SED97] E.L. Sonnhammer, S.R. Eddy, and R. Durbin. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28(3):405–420, July 1997.

[SEM98] J. Stoye, D. Evers, and F. Meyer. Rose: generating sequence families. *Bioinformatics*, 14(2):157–163, January 1998.

[SH05] V.A. Simossis and J. Heringa. PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information. *Nucleic acids research*, 33(Web Server issue):W289–W294, July 2005.

[SHS$^+$10] A.R. Subramanian, S. Hiran, R. Steinkamp, P. Meinicke, E. Corel, and B. Morgenstern. Dialign-tx and multiple protein alignment using secondary structure information at gobics. *Nucleic Acids Research*, 38(Web-Server-Issue):19–22, 2010.

[SM58] R.R. Sokal and C.D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Scientific Bulletin*, 28:1409–1438, 1958.

[SM11] E.F. Sandes and A.C.M.A Melo. Smith-waterman alignment of huge sequences with gpu in linear space. In *Parallel Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pages 1199 –1211, may 2011.

[SMD97] J. Stoye, V. Moulton, and A.W. Dress. DCA: an efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. *Computer applications in the biosciences : CABIOS*, 13(6):625–626, December 1997.

[SMKM05] A. Subramanian, J.W. Menkhoff, M. Kaufmann, and B. Morgenstern. DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6(1):66+, 2005.

[SN87] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular biology and evolution*, 4(4):406–425, July 1987.

[SNKM04] M. Schmollinger, K. Nieselt, M. Kaufmann, and B. Morgenstern. Dialign p: Fast pair-wise and multiple sequence alignment using parallel processors. *BMC Bioinformatics*, 5:128, 2004.

[SS91] C. Sander and R. Schneider. Database of homology derived protein structures and the structural meaning of sequence alignment. *Proteins: Struct. Funct. Genet.*, 9:56–68, 1991.

[SW81] T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, March 1981.

[SWD$^+$11] F. Sievers, A. Wilm, D. Dineen, T.J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J.D. Thompson, and D.G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1), October 2011.

[Tay86] W.R. Taylor. Identification of protein sequence homology by consensus template alignment. *Journal of Molecular Biology*, 188(2):233–258, March 1986.

[Tay88] W.R. Taylor. A flexible method to align large numbers of biological sequences. *Journal of Molecular Evolution*, 28:161–169, 1988.

[TGP$^+$97] J.D. Thompson, T.J. Gibson, F. Plewniak, F. Jeanmougin, and D.G. Higgins. The CLUSTAL_X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools. *Nucl. Acids Res.*, 25(24):4876–4882, December 1997.

[THG94] J.D. Thompson, D.G. Higgins, and 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(22):4673–4680, November 1994.

[TMX$^+$11] P.D. Tommaso, S. Moretti, I. Xenarios, M. Orobitg, A. Montañola, J-M.M. Chang, J.F. Taly, and C. Notredame. T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension. *Nucleic acids research*, 39(Web Server issue):W13–W17, July 2011.

[TOG$^+$10] P.D. Tommaso, M. Orobitg, F. Guirado, F. Cores, T. Espinosa, and C. Notredame. Cloud-Coffee: Implementation of a parallel consistency-based multiple alignment algorithm in the T-Coffee package and its benchmarking on the Amazon Elastic-Cloud. *Bioinformatics*, 26(15):1903–1904, 2010.

[TPP99] J.D. Thompson, F. Plewniak, and O. Poch. BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, 15(1):87–88, January 1999.

[TPR$^+$01] J.D. Thompson, F. Plewniak, R. Ripp, J.C. Thierry, and O. Poch. Towards a reliable objective function for multiple sequence alignments1. *Journal of Molecular Biology*, 314(4):937–951, December 2001.

[TS09] C. Trapnell and M.C. Schatz. Optimizing data intensive gpgpu computations for dna sequence alignment. *Parallel Computing*, 35(8-9):429–440, 2009.

[VA91] M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, 218:33–43, 1991.

[Vit67] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

[VS93] M. Vingron and P.R. Sibbald. Weighting in sequence space: a comparison of methods in terms of generalized sequences. *Proc Natl Acad Sci U S A*, 90(19):8777–81, 1993.

[VW94] M. Vingron and M. Waterman. Sequence alignment and penalty choice: Review of concepts, case studies and implications. *Journal of Molecular Biology*, 235(1):1–12, January 1994.

[Wal96] M. Wall. GAlib: A C++ library of genetic algorithm components. *Mechanical Engineering Department, Massachusetts Institute of Technology*, 1996.

[WHN08] A. Wilm, D.G. Higgins, and C. Notredame. R-Coffee: a method for multiple alignment of non-coding RNA. *Nucleic acids research*, 36(9), May 2008.

[WJ94] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of computational biology : a journal of computational molecular cell biology*, 1(4):337–348, 1994.

[WLW05] I.V. Walle, I. Lasters, and L. Wyns. SABmark–a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics (Oxford, England)*, 21(7):1267–1268, April 2005.

[WOHN06] I.M. Wallace, O. O'Sullivan, D.G. Higgins, and C. Notredame. M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Research*, 34(6):1692–1699, 2006.

[WSH08] K.M. Wongaren, M.A. Suchard, and J.P. Huelsenbeck. Alignment Uncertainty and Genomic Analysis. *Science*, 319(5862):473–476, January 2008.

[XM04] W. Xu and D.P. Miranker. A metric model of amino acid substitution. *Bioinformatics*, 20(8):1214–1221, 2004.

[ZYRA07] J. Zola, X. Yang, A. Rospondek, and S. Aluru. Parallel-tcoffee: A parallel multiple sequence aligner. In Ghulam Chaudhry and Soo-Young Lee, editors, *ISCA PDCS*, pages 248–253. ISCA, 2007.

Acknowledgements/Credits

- Miquel Orobitg is funded by the CUR of DIUE of GENCAT.
- This thesis was supported by the Ministry of Education and Science of Spain under contract TIN2011-28689-C02-02, TIN2010-12011-E and Consolider CSD2007-00050.
- This thesis has received a grant for linguistic revision from the Linguistic Service at the University of Lleida (2013 call).