High Performance Computing Improvements on Bioinformatics Consistency-Based Multiple Sequence Alignment Tools

Miquel Orobitg[a], Fernando Guirado[a], Fernando Cores[a], Jordi Llados[a], Cedric Notredame[b]

[a] *Universitat de Lleida, EPS Building, Avda. Jaume II, n.69, 25001 Lleida, Spain*
[b] *Universitat Pompeu Fabra - Centre de Regulacio Genomica (CRG), PRBB Building, Dr. Aiguader, n.88, 08003 Barcelona, Spain*

Abstract

Multiple sequence alignment (MSA) is one of the most useful tools in bioinformatics. MSA plays a key role in protein/RNA structure prediction, phylogenetic analysis or pattern identification among other important bioinformatic applications. However, the growth of sequencing data imposes further difficulties to aligning it with traditional tools. For large-scale alignments with thousands of sequences or even whole genomes, it will be necessary to use and take advantage of high performance computing (HPC). This paper, focused on a consistency-based MSA tool called T-Coffee, presents several innovative solutions; the Balanced Guide Tree, the Optimized Library Method and the Multiple Tree Alignment. The results obtained by the methods presented have demonstrated that it is possible to improve efficiency, scalability and accuracy.

Keywords: Multiple Sequence Alignments, Consistency, T-Coffee,High Performance Computing, Scalability, Accuracy

Email addresses: `miquel.orobitg@crg.eu` (Miquel Orobitg), `f.guirado@diei.udl.cat` (Fernando Guirado), `fcores@diei.udl.cat` (Fernando Cores), `jordi.llados@udl.cat` (Jordi Llados), `cedric.notredame@crg.eu` (Cedric Notredame)

1. Introduction

Due to the advent of the era of comparative genomics, it has become increasingly important to be able compare a large number of homologous sequences simultaneously [17]. Sequence Alignment, particularly Multiple Sequence Alignment (MSA), is of central importance to bioinformatics and computational biology [8]. The main idea behind MSA is to put protein residues in the same column according to selected criteria. These criteria can be the structural, evolutionary, functional or sequence similarities.

Accurate and fast construction of multiple sequence alignments has been extensively researched in recent years. There are many MSA heuristics, and the progressive alignment method is one of the most widely used [3]. However, the exponential growth of biological data and the inability to treat them efficiently have led to many of the existing methods becoming obsolete through not being able to align thousands of sequences. This problem has highlighted the need for biologists, bioinformatics and computer scientists to work jointly. Thus, some MSA methods introduce High Performance Computing capabilities to take advantage of the new technologies and infrastructures.

MSA tools have exhibited scalability problems when the number of sequences increases. The authors' research is focused on allowing such tools to take advantage of HPC systems and so improve their efficiency, scalability and accuracy. In the present paper, the study is based on the T-Coffee MSA tool [10]. T-Coffee is a multiple sequence alignment package that allows DNA, RNA, protein sequences and structures to be aligned. Thus, in order to improve the efficiency and scalability of T-Coffee, this paper proposes the integration of 3 different solutions:

- The Balanced Guide Tree method (BGT). The multiple sequence alignment process follows the order given by a guide tree that identifies the closely related sequences and their alignment precedences. The guide tree structure exhibits parallel optimizations in the alignment process. The proposed BGT method is capable of overcoming the initial degree of parallelism by taking advantage of increasing computer resources.

- The Optimized Library Method (OLM). Consistency-based MSA tools store the sequence relationship in a dedicated library. The more sequences that are aligned, the more data is stored, the larger the amount of memory needed and processing time also increases. The OLM is a

new library-building method able to optimize the consistency library, thus allowing bigger sets of sequences to be processed and reducing the execution time.

- The Multiple Tree Alignment process (MTA). The guide tree is defined by heuristic methods that try to obtain the best accuracy. However, determining the best guide tree is a difficult challenge and a slight variation in the guide tree could produce a better result. The MTA provides different variations of any guide tree, preserving the original criteria of aligning first the most related sequences. Then, it processes all of them in parallel and finally trys to select the most accurate alignment.

The rest of the paper is organized as follows: Section 2 presents a brief state of the art of MSA tools. In Section 3 some of the scalability problems of T-Coffee MSA tool are analyzed. In Section 4, we present our approaches to solving the previously presented problems. The performance, scalability and accuracy evaluation are shown in Section 5 and finally the main conclusions are presented in Section 6.

2. State of Art

The computation of an optimal mathematical alignment is an NP-Complete problem [23]. For this reason, current implementations of the MSA algorithms are heuristic and none of them guarantees full optimization. The progressive alignment is one of the most widely used heuristic. It assembles a multiple alignment by making a series of pairwise alignments of sequences, which are added one by one following the order established by a guide tree. The most popular progressive alignment implementation is the Clustal family [20].

Although this heuristic provides a great advantage in 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 performing a progressive alignment and then refining the result by repeatedly dividing the aligned sequences into sub-alignments and realigning the sub-alignments. The most relevant iterative aligners are MAFFT [5], Muscle [15] and ClustalΩ [18].

Consistency-based methods were designed to overcome the accuracy limits caused by the greediness problems of progressive and iterative aligners, using sequences information to avoid the mistakes in the alignment. The common idea of consistency-based approaches is to evaluate pairwise alignments through the comparison of third sequences. This consistency information can then be used to construct the alignments or evaluate them, depending on the approach. However, the introduction of more information increases the memory requirements. The first combination of a consistency-based scoring scheme with a progressive alignment algorithm was described by Notredame in T-Coffee [10]. Other common MSA programs that use consistency to produce accurate alignments are Probcons [2], Probalign [16], and Dialign [9].

T-Coffee (TC), is a multiple sequence aligner method that combines the consistency-based scoring function COFFEE [11] with the progressive alignment algorithm. TC introduces a library generated using all-against-all pairwise alignments computed with a pair of Hidden Markov Models (HMM) in order to reduce the greediness and increase the accuracy compared with other methods based on a progressive strategy.

In spite of the improvement in speed introduced by the heuristics, the computational requirements for large-scale alignments (thousands of sequences) clearly exceed workstation performance. Therefore, parallel implementations based on the main heuristics, such as ClustalW-MPI [6], Parallel-TCoffee [24] or DialignP, were implemented. More recently, different approaches have used graphics processing units (GPUs) to reduce the execution time of MSA applications [4]. The development of MSA-CUDA [7] or MUMmerGPU [22] are examples of such applications. Although all of these approaches improve their original algorithm, these methods exhibit scalability problems when the number of sequences increases. These are due to data dependencies and memory requirements.

3. T-Coffee Scalability Issues

T-Coffee provides an improvement in accuracy over most methods based on a progressive strategy. However, the introduction of these improvements has penalized T-Coffee in speed when it is compared with the most commonly-used alternatives.

The overhead introduced by the consistency-based scheme can be understand if we analyze the T-Coffee algorithm, presented in Figure 1. As can be

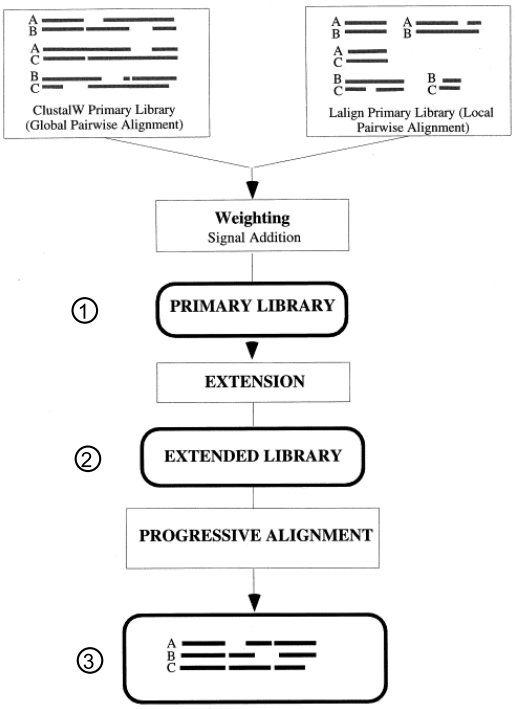


Figure 1: T-Coffee algorithm

seen, T-Coffee is divided into three main steps:

1. **Primary Library**. The primary library contains a set of pairwise alignments from among all the sequences to be aligned. In the library, each alignment is represented as a list of pairwise residue matches (constraints). 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.
2. **Extended Library**. 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.
3. **Progressive Alignment strategy**. The MSA is produced by the progressive alignment strategy introduced in Section 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 a profile pairwise alignment technique and maximizing the COFFEE objective function using the weights in the extended library instead of using the substitution matrix weights and gap penalties.

The increasing complexity of the progressive alignment, added to the rising computing cost from the generation of all pairwise alignments needed 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.

Figure 2 analyzes the execution time depending on the number of sequences. It can be seen that runtime requirements grows quadratically with the number of sequences extracted from the PFAM, a protein family database [19]. As a result, T-Coffee is incapable of aligning more than 200 sequences of 500 residues on a standard desktop computer.

The parallelization of TC is a manner to increase its performance and scalability. This way, the latest versions of TC have parallelized some parts to take advantage of the shared-memory multi-core architectures [21]. This new concurrent version is capable of reducing the execution time. However, the problem of the TC scalability remains due to the high memory requirements to maintain the consistency-based library.

In the following subsections, we present the TC analysis from the point of view of its parallelism capabilities and the memory resources usage in order to clarify the way in which our proposed methods are applied.

3.1. Parallelism Performance Analysis

In this section, we study the T-Coffee performance and scalability features. Our objective is to detect possible bottlenecks in the scalability of T-Coffee or any of its five different stages. Thus, we analyze the evolution of execution time for each step of the T-Coffee algorithm when more processors are used.

The analysis, which was done using the TC parallel implementation (PTC) [24], varied the numbers of processor from 16 to 120, to align the PF00231

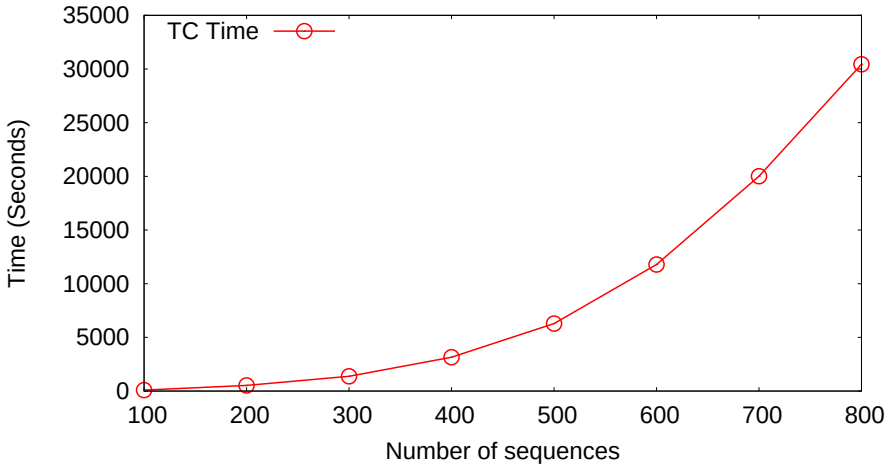


Figure 2: T-Coffee execution time analysis

sequence set from the Pfam database. This is made up of 554 sequences and a maximal length of 331 amino-acids.

Figure 3 shows the total execution time and the five main stages of the TC algorithm (Initialization, Distance Matrix, Primary Library calculation, Extension of the Consistency Library and the Progressive Alignment) in function of the number of processors. As can be observed, PTC can reduce the TC execution time. However, with more than 64 processors, the speedup stagnates, limiting the scalability of PTC.

With regards to the PTC stages, the most time-consuming steps are the calculation of the Primary Library and the Progressive Alignment. These two stages consume more than the 98% of the execution time. The library generation shows good scalability while the progressive alignment stage remains linear up to 48 processors. This 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. However, it is known that the progressive alignment stage is bounded by task dependencies extracted from the guide tree, thus limiting its scalability.

3.2. Memory Constraint

In TC, consistency is achieved through a collection of pairwise alignments called library, L, and represented by a NxN matrix, N being the number of sequences and position $L(i, j)$ is a list of a pairwise residue matches for sequences i and j ($i \neq j$), called *constraints*. Each constraint is a 3-tuple

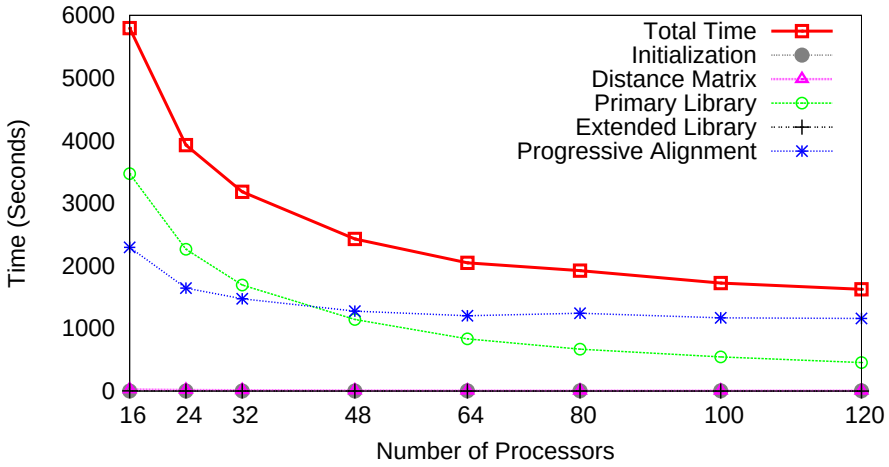


Figure 3: T-Coffee stage execution time

$\{S_i^x, S_j^y, W_{i,j}\}$, where S_i^x denotes 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 used to identify the correctness of a constraint in the *sequence identity*. TC assigns this weight to each constraint in order to give more priority to the most reliable residue pairs.

The size of the library is $N^2 * l$, where l is the average length of the input sequences. Figure 4 shows the growth of memory based on N and l. The memory requirements grow quadratically, turning TC into a non-scalable method incapable of aligning large numbers of long sequences.

To deal with this problem, TC incorporates a new library generation method that reduces memory requirements by using BLAST [1] to build it from a subset of sequences, instead of doing 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, this approach reduces the quality of the alignments considerably.

4. MTA-TCoffee (Multiple Tree Alignment - TCoffee)

In this section, we present our method, called MTA-Coffee, which integrates different solutions to lessen the existing MSA parallelization problems. These solutions deal with the limited parallelism in the progressive alignment stage and the high memory requirements of the consistency based methods

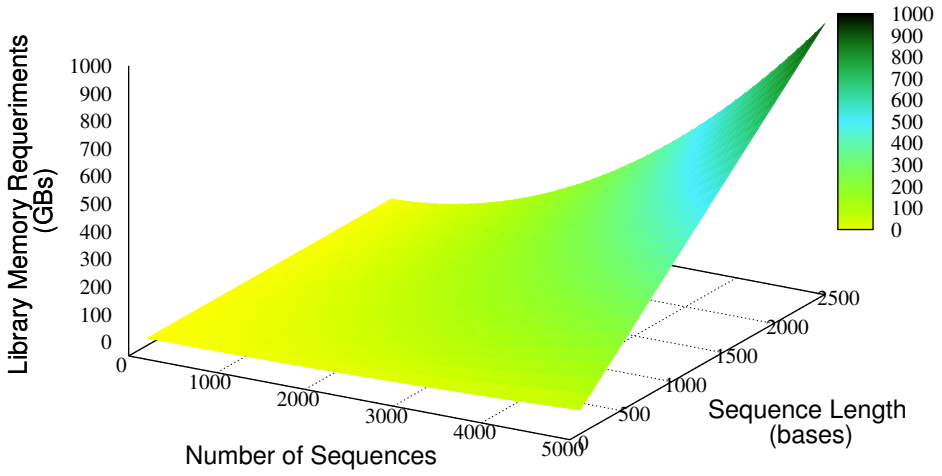


Figure 4: Analysis T-Coffee memory requirements.

without loss of accuracy in the final result.

4.1. Balanced Guide Tree (BGT)

As shown above, what limits the speedup on TC is the progressive alignment stage. This stage is driven by the neighbor-joining guide tree (NJ), which fixes the order of the partial alignments in the progressive alignment. Therefore, the guide tree is the key that defines the dependences among these tasks.

Figure 5 shows a guide tree generated by the NJ algorithm. The PT-nodes define the progressive alignment tasks. The leaf nodes are the sequences to be aligned and the tree represents the order in which such progressive alignments must be performed. Only PT-nodes with all dependencies resolved can be executed as independent tasks. In the example, there are three initial tasks, grey PT-nodes, which can be launched in parallel. The critical path defines the sequential iterations that the algorithm has to perform. The more sequential iterations that must be done, the lower the parallelism, the lower the performance and the higher the execution times. In the example, the maximum degree of parallelism is three tasks and the critical path length is 7 tasks, thus requiring 7 sequential iterations to complete the alignment.

We evaluated some of the NJ guide trees from the Pfam database to determine the parallelism features. The evaluation shows that NJ builds

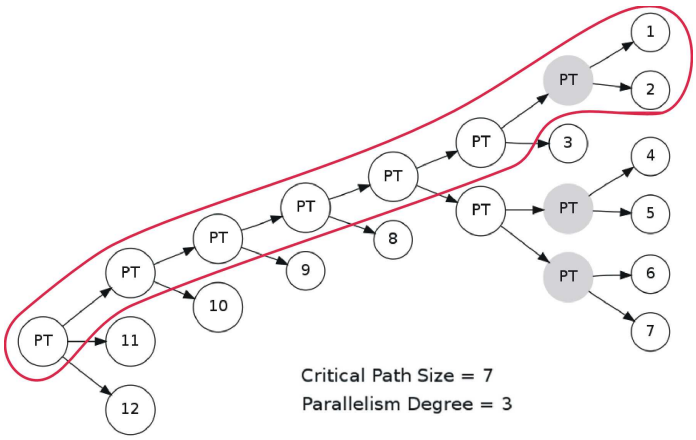


Figure 5: Guide tree generated with standard NJ heuristic

are very unbalanced guide trees with long critical paths and low degrees of parallelism. In [14], the authors proposed a new tree generation method called Balanced Guide Tree (BGT) that reduces the critical path and obtains a higher degree of parallelism, without losing accuracy.

The BGT algorithm is derived from the original NJ algorithm with the aim to maintain the high-related-sequences-first criteria and also balancing the guide tree. Thus, BGT tries to join the maximum number of pairs of sequences locating them at the base of the tree in order to reduce the number of tree levels and then reducing the critical path. It is important to note that BGT evaluates the similarity of the sequences and determines if they can be re-allocated but with the aim of maintaining the alignment accuracy.

Table 1 compares the critical path (CP) and maximum degree of parallelism (MPD) of the original NJ clustering algorithm with those obtained by the BGT method. The results show that, on average, BGT reduces the critical path by 42.8% and increases the degree of parallelism by 84.2%.

Thus, we can conclude that by using the balanced guide trees obtained it may be possible to take advantage of parallel computing infrastructures given that they can execute more parallel tasks.

4.2. Optimization Library Method (OLM)

To reduce the memory requirements of T-Coffee we propose a new library building method, called OLM, which optimizes the library size by reducing the consistency data stored [12]. This reduction means lower execution times

	Critical path (CP)			Max Parall (MPD)		
Sequences	**NJ**	**BGT**	**Δ**	**NJ**	**BGT**	**Δ**
PF00074	28	11	-25.0%	107	172	60.7%
PF00349	23	19	-17.3%	143	252	73.2%
PF00231	26	18	-30.7%	164	270	64.6%
PF01057	94	14	-85.1%	71	187	163%
PF00200	29	17	-41.3%	173	295	70.5%
PF08443	68	29	-57.3%	215	365	69.7%
Average			-42.8%			84.2%

Table 1: NJ and BGT comparison for the Pfam database

and allows TC to handle a large number of sequences.

The Library optimization is based on two complementary methods. The first of these, the Essential Library Method, is applied to the Primary Library construction and 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. It 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 1, interprets the sequence identity weight of a constraint (line 5) and compares it against other constraints from the same residue in the library (line 6). If the constraint provides more accurate consistency information, then it replaces the existing one in the library.

The second method, called Threshold Library, discards the constraints that provide little or no information for the alignment. It identifies the constraints with some influence on the alignment and evaluates the deviation of their weight with regard to the maximum in the library. The user defines the maximum allowed deviation, i.e. *threshold*. The residues with values lower than the threshold are discarded. The threshold determines how aggressive the reduction of the library can be. The more aggressive the reduction of the library, the faster it can be built, the bigger the dataset can be computed, but less accurate the alignments that will be obtained. Conversely, the smaller the reduction, the slower it is to build and the bigger, but more accurate, the alignments will be.

Algorithm 1 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}=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. $L(S_i^x, S_j^y) = \max(L(S_i^x, S_j^y), W_{(x,y)})$
7. end_for
8. end_for
9. end_for

Both methods, Essential and Threshold Library, can be applied together. In Algorithm 2, 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 its greater, the same criterion will be applied as in the Essential Library Method (line 7).

Algorithm 2 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

This optimization of the library is done during its generation and not after. On the other hand, if the optimization were done when the library has already been built, the huge amount of data stored in the library would

saturate the memory system and the alignment could not be completed. This way, it is possible to reduce the time cost of building the library and also it allows a smaller library to be obtained, which is able to store the consistency data needed to compute bigger alignments.

In order to determine the impact of the threshold parameter, we conducted an experimental study over the alignment accuracy and also the execution time, applying threshold values from 10% to 80%. Table 2 shows the accuracy, library size and execution time for the original T-Coffee and eight OLM approaches applying also the Essential library reduction in all cases. For this experimentation, we used all sequence sets from the PREFAB database [15].

Aligner	**Acc.**	**Lib Size (MB)**	**Time (s)**
TC	0.709	1,642.60	1,646
OLM TC-Lib.10	0.700	649.60	1,216
OLM TC-Lib.20	0.698	613.36	1,210
OLM TC-Lib.30	0.695	579.98	1,198
OLM TC-Lib.40	0.692	550.60	1,190
OLM TC-Lib.50	0.689	523.42	1,184
OLM TC-Lib.60	0.686	497.86	1,176
OLM TC-Lib.70	0.675	474.80	1,171
OLM TC-Lib.80	0.661	451.71	1,163

Table 2: Comparison between different OLM TC-Library configurations varying the level of library optimization

It can be observed that the best results are obtained with the *OLM TC-Library10* with obtains similar accuracy to TC but is 26.12% faster. With regards to the library, the memory requirements are drastically reduced (from 1,642MB to 649MB). Also, the cut-off maintains linear behavior with the threshold (each 10% decreases the library by 27 MB). We selected the *OLM TC-Library10* as a good compromise between memory requirements, accuracy and execution time, and also the *OLM TC–Library50* as a more aggressive configuration in order to evaluate the effectiveness of using higher library reductions.

To sum up, we can conclude that the OLM is able to reduce the size of the consistency library, which has a very important impact on further opti-

mizations that increase the TC scalability and which would not be possible any other way.

4.3. Multiple Tree Alignment (MTA)

The guide tree determines the accuracy of the final alignment. Slight modifications to it can produce different results. Thus, it is possible to improve the final alignment accuracy by applying little modifications to the guide tree. However, determining the modifications that must be applied is an important challenge as these must reduce the noise generated by possible bad alignments performed in the first iterations of the algorithm.

To reveal the potential of this approach for increasing accuracy of T-Coffee and to quantify the error introduced by the guide tree, an experiment was conducted in which the best guide tree could always be selected by using a benchmark score. The experiment (Figure 6) shows the evolution of the alignment accuracy depending on the number of trees treated (up to 300 trees). It uses the whole PREFAB datasets as the input sequences and the Q score as the validation score to measure the improvements in accuracy. The Q Score is also used as the selection metric to determine the best alignment.

Figure 6 indicates that the average alignment accuracy rises as the number of trees analyzed increases. Specifically, it is shown that, evaluating 300 trees, the Q accuracy improves from 0.709 to 0.759. It is noteworthy that when only one guide tree is used, the first graph value corresponds to the accuracy obtained by the default T-Coffee.

These results show that the error introduced by the guide tree is significant and could be mitigated if more trees were evaluated. It is known that benchmarking scores cannot be used because they are based on reference alignments, which are not always available. However, the use of this benchmarking scores is useful for validating the new method, because they allow the maximum degree to which the quality of the alignments can be improved to be determined, depending on the number of trees evaluated.

To cope with the errors from the progressive alignment due to the guide tree, we propose the Multiple Tree Alignment (MTA). MTA is a new method implemented in T-Coffee that consists of creating, aligning and evaluating multiple guide trees in parallel, in order to improve the biological accuracy of the alignment. The MTA method can be applied to any progressive aligner that accepts guide trees as an input parameter. MTA is capable of improving the accuracy of these aligners without modifying the original methods.

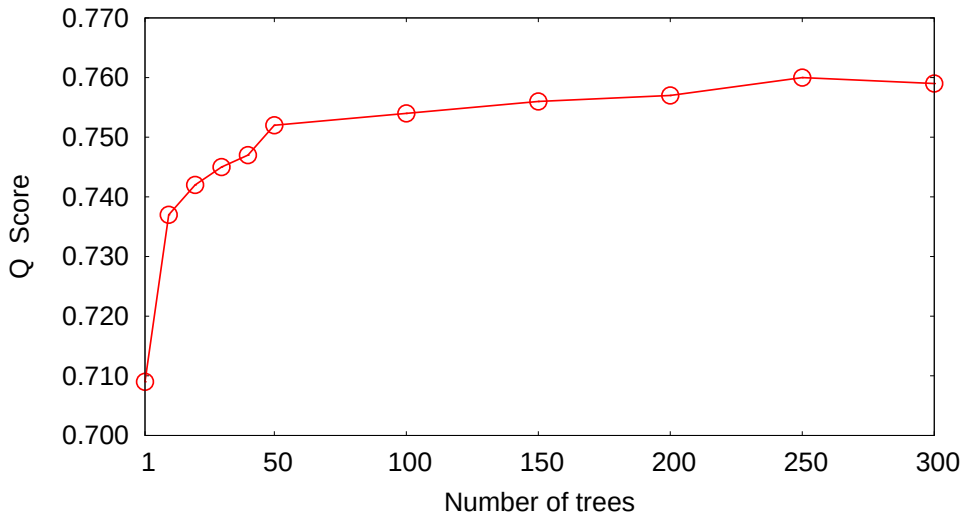


Figure 6: Multiple tree validation using the Q benchmark score as the evaluation metric

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. This is because it is possible that the best computational match alignment does not exhibit the best biological meaning. Thus, we carried out an extensive comparison study to analyse different evaluation metrics in order to decide which one provides a better correlation between the computational score and biological quality.

In order to apply the MTA method, two additional steps were introduced into the T-Coffee algorithm:

- **Guide Tree Generation step.** During this step, the method produces N different guide trees based on the NJ clustering algorithm, N being defined by the user. The variation introduced in the guide tree is low enough to keep the distance criteria but significant enough to provide the necessary flexibility to generate multiple alternative trees.

- **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 MTA method. This method allows the user to choose among different external evaluation functions. MTA is capable of evaluating the accuracy of the alignments with the

following single scores: SP, NoRMD, COFFEE, TRIPLET, iRMSD and STRIKE. The authors also propose the use of two meta-scores obtained through genetic algorithms: the Weighted-Score chromosome (WSC-GA) and the Meta-score Code Chromosome (MCC-GA) [13]. The main idea of these meta-scores is to combine the main characteristics of the various metrics and thus, finding more accurate alignments.

4.4. Paralellization Approach

From version 8.0 onwards, TC has been parallelized using a multi-process implementation taking advantage of shared-memory multi-processor/multi-core architectures [21]. However, this parallelization has two main problems: 1) The poor degree of parallelism on the progressive alignment stage; 2) The consistency library memory requirements that restrict TC to aligning a small number of sequences on a single machine.

We propose the BGT and OLM methods to solve these problems. The BGT increases the degree of parallelism, improving the efficiency of T-Coffee, while the OLM method is dedicated to enhancing its scalability. And finally, we propose the use of the MTA method to minimize the impact on the accuracy of the original method. The tool resulting from the integration of all the proposed techniques is called MTA-TCoffee.

In order to design our proposed MTA-TCoffee, it is firstly necessary to determine the parallel infrastructure to be used. The proposal is based on two main architectures: a distributed architecture based on a cluster of workstations where each one of these corresponds to a shared-memory multicore. The proposed design is presented in Figure 7.

In such an infrastructure, the MTA-TCoffee assign to each multicore workstation a single guide tree that it is processed in a parallel manner, applying the BGT method. In this situation, the OLM is applied to reduce the total amount of memory required to maintain the consistency library, and thus it is possible to increase the number of sequences to be aligned. The accuracy quality parameter is ensured as the MTA allows multiple variations of the guide tree to be processed and the best one selected. As all guide trees are processed concurrently, the total execution time is similar to the needed to process only one.

5. Experimentation

In this section we present the analysis of the performance, scalability and accuracy of MTA-TCoffee proposal. In the experimentation we compare the

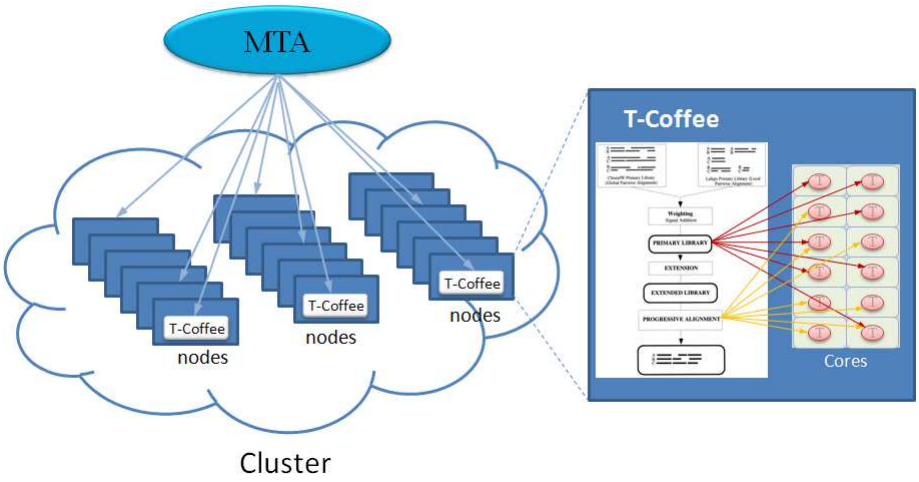


Figure 7: Parallelization of MTA-TCoffee.

performance of the original T-Coffee (TC) with the MTA T-Coffee without OLM (MTA-TC) and the MTA-TC after applying two configurations of the OLM method, with the library reduction of 10% and 50%(MTA-TC T-Library10 and MTA-TC T-Library50 respectively). Furthermore, MTA is configured to build 95 guide trees.

The parallel infrastructure is based on a cluster composed of 12 computing nodes. Each computing node contains two 2.4GHz Intel quad cores and 8Gb of RAM, giving a total of 96 cores. The interconnection network is a Gigabit Ethernet.

5.1. Performance Analysis

In this section, we study the execution times of the MTA-TCoffee, comparing it with the original method using a prefabricated 200-sequence set from Pfam. Naturally, the serial version of MTA cannot compete with original aligners on executing time, because the trees have to be aligned serially. 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.

The main goal is to determine the ability of our proposal to reduce its execution time while increasing the number of assigned process units. To do so, we increased the number of assigned cores from 8 to 96, one being

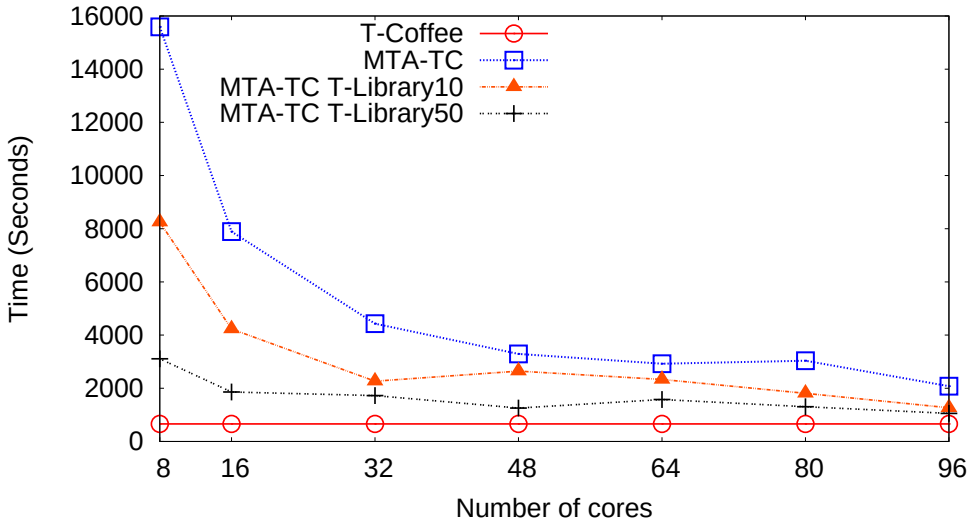


Figure 8: MTA-TCoffee execution time

the master that maintains the consistency library. All configurations require the use of the interconnection network to avoid any advantage from the architecture.

Figure 8 compares the parallel MTA-TC T-Library10, T-Library50 against the MTA-TC (without any library reduction) and the standard T-Coffee. As can be observed, the execution times of the MTA-TC T-Library10, MTA-TC T-Library50 and MTA-TC decrease when the number of processors increases. It is also observed that the library reduction has a great impact on the final execution time. Thus, with 8 cores, the MTA-TC needs 15,905 seconds, while the MTA-TC T-Library10 takes 8,415 sec. (47.09%) and the MTA-TC T-Library50, 3,171 (80.06%) sec. When the number of cores increases, the execution time also decreases, being 39.15% and 49.24% respectively. However the differences narrow due to the bottleneck of access to the centralized consistency library. Finally, it is also observed that our proposals are slower than the original TC, independently of the number of processors used because of the network contention access to the centralized library.

To analysis the impact of the BGT method, we performed another experiment comparing the BGT and the original Neighbor-Joining (NJ) methods. In this case, we did not use any library reduction, the MTA-TC being the selected method.

The results obtained are shown in Figure 9. It can be noted that the

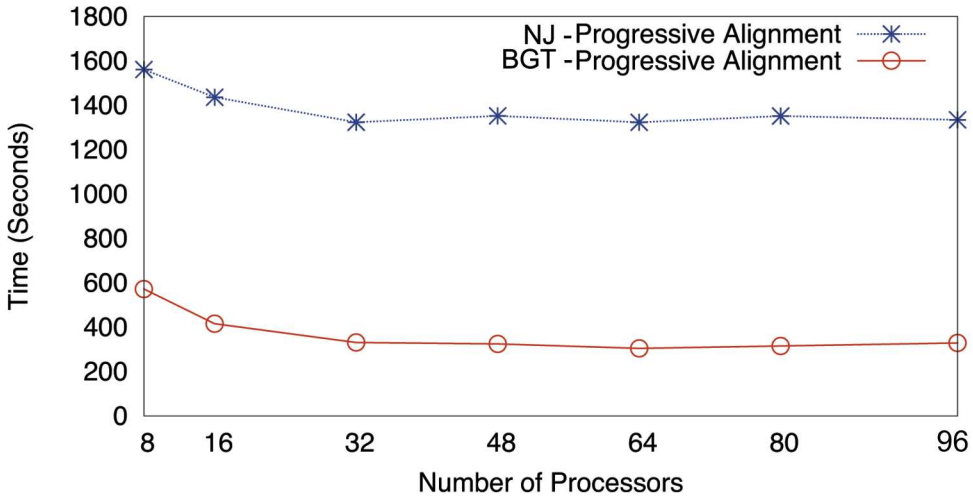


Figure 9: BGT impact on the execution time

NJ guide three not scale well due to its very unbalanced structure, which limits the number of parallel alignment tasks that can be performed. This behaviour can be explained by the fact that as new processors are added, only the first iterations of progressive alignment are able to take advantage. After these first iterations, the degree of parallelism decreases quickly, with an increasing number of processors being kept idle. Instead of this behavior, the BGT method drastically reduced the execution time by a 62% (from 1587 to 600 sec.), obtaining a better scalability. However, this is also limited by the fact that even if the tree is perfectly balanced, the number of scheduled task is halved at each iteration, and after a few iterations, there are not enough tasks for fill all the processors.

Finally, the main conclusion of this experimental study is that, by using the OLM library reduction combined which the BGT method, our proposal is able to reduce the execution time of the progressive alignment.

5.2. Scalability Analysis

In this section, the scalability problem of our proposal, MTA-TCoffee, was analysed and compared with TC. The experiment consisted of increasing the number of sequences to be aligned, and then analysing how the performance and resource requirements behaved.

The experiments consisted of running first TC on a node of the cluster, and then, launching MTA-TC T-Library10 and T-Library50 in parallel. In

order to facilitate the comparison with TC, for MTA we only took into consideration the memory requirements and the execution time for aligning one guide tree.

Figures 10a and 10b show the evolution of the library memory requirements and the total execution time on increasing the number of aligned sequences from the Pfam database 100 to 2000. In Figure 10a the TC and the MTA-TC approaches are compared. It can be observed that with 900 sequences, the library size is reduced by 69.31% and 82.24% respectively. In this experimental study TC is unable to align more than 900 sequences due to the huge amount of memory needed to maintain the consistency library. In contrast, both the MTA-TC approaches can align more than 2,000 sequences. This study demonstrates that the MTA-TC is more scalable than TC as it could align up to twice as many sequences. The behavior of the MTA-TC T-Library10 and T-Library50, are similar, growing more slowly than the original TC, and needing less memory in the case of the T-Library50.

Analyzing the execution time, Figure 10b shows similar behavior, where both approaches in MTA-TC are faster than TC, which is impractical over 900 sequences. In this study, the MTA-TC T-Library10 runs slower than MTA-TC T-Library50 due to the fact that the access to its larger consistency library is slower.

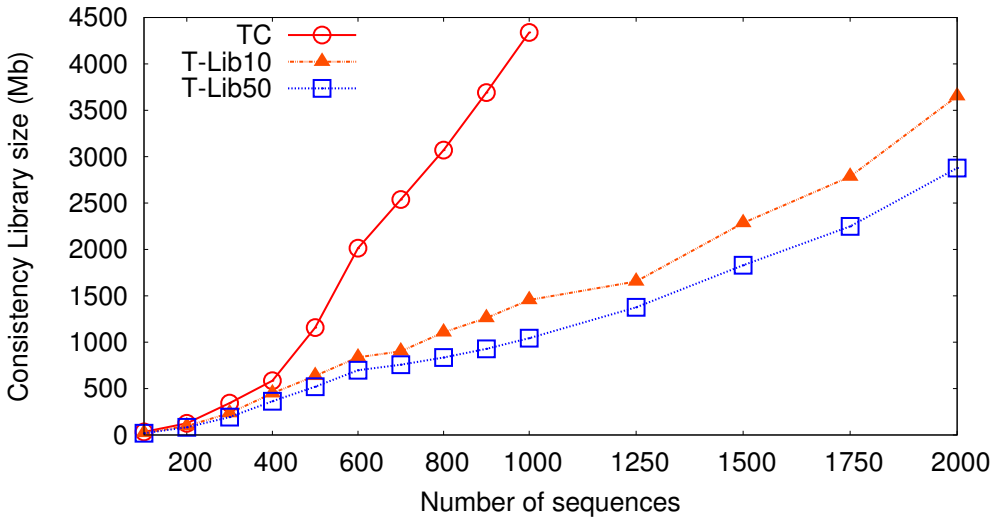
5.3. Accuracy Analysis

In this section, we analyze how the number of trees affects the alignment quality and compare the accuracy of MTA-TCoffee with other aligners.

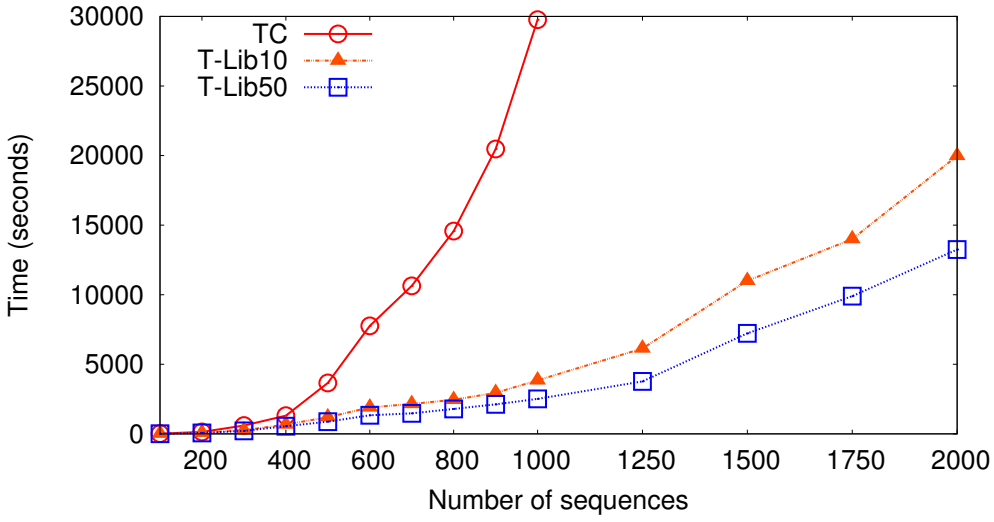
5.3.1. Impact of the number of guide trees

Figure 11 shows the evolution of the alignment accuracy with the number of trees treated by the MTA. The experiment evaluates up to 100 trees using the whole PREFAB datasets as the input sequences, the Q score as the validation score to measure the improvements in accuracy and NiRMSD score as the selection metric. We also added the T-Coffee accuracy obtained to emphasize MTA's ability to improve the final result.

As the theoretical results suggested, Figure 11 indicates that, in all the MTA-TC configurations, the average alignment accuracy rises as the number of trees analyzed increases. Specifically, it is shown that when evaluating 100 trees, MTA-TC is able to improve the Q accuracy from 0.709 to 0.731. If Figure 11 is analyzed in more detail, it can be seen that the largest increase in the accuracy of alignments is achieved with 10 trees. Then, the quality



(a) Memory requirements analysis



(b) Total execution time analysis

Figure 10: MTA-TCoffee sequence scalability study

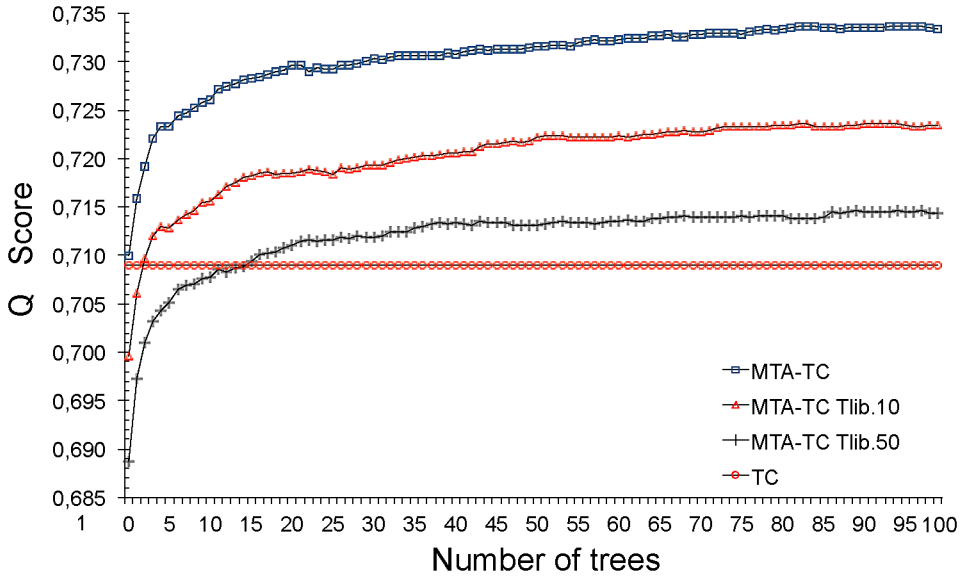


Figure 11: Accuracy varying the number of trees

grows progressively up to 20 trees, but from then on, the improvement is smaller, at the same time that the number of trees analyzed increases. These results demonstrate that the MTA-TC method is able to improve the quality of an alignment obtained from an MSA method by selecting a better guide tree.

We should also note that the inclusion of MTA lets us more than regain the quality lost due to the library reduction (OLM). Thus, both the MTA-TC T-Library10 (0.721) and MTA-TC T-Library50 (0.713) are able to recover and exceed the quality obtained by the original method (0.709)

5.3.2. MSA Applications Comparison

The main aim of the last experimental was to compare the alignment accuracy obtained with MTA-TC proposal against some of the most common consistency-based, iterative or progressive-alignment MSA applications. To do it, we used the MTA-TC without any library reduction, and both the MTA-TC T-Library10 and T-Library50 cases studies. The selection metric used was the NiRMSD score. 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 obtained results are shown in the Table 3,

where the rows identify the method compared and the columns identify the range of identity. The best results obtained are highlighted in bold.

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

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

Aligner	**0-15**	**15-25**	**25-35**	**35-100**	**Avg.**
MSAProbs	0.454	**0.756**	**0.900**	0.961	**0.738**
MTA-TC	**0.470**	0.743	0.885	0.953	0.731
MAFFT	0.431	0.743	0.887	0.958	0.724
MTA-TC T-Lib.10	0.449	0.733	0.878	0.946	0.720
Probalign	0.424	0.732	0.891	0.962	0.719
MTA-TC T-Lib.50	0.428	0.723	0.877	0.948	0.711
T-Coffee	0.421	0.721	0.876	0.951	0.709
ClustalΩ	0.395	0.708	0.878	**0.965**	0.700
Muscle	0.365	0.684	0.860	0.951	0.677
ClustalW	0.289	0.605	0.816	0.941	0.617

Table 3: Comparison of the accuracy of the MSA methods.

6. 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 NP-Complete problem, whose solution stands at a crossroads between biology and computation.

The present paper shows the effectiveness of the integration of three different methods focused on allowing the use of the high-performance computing capabilities of the consistency-based multiple sequence alignment tools used by Bioinformatics. These methods are:

- The Balanced Guide Tree (BGT), which is devoted to solving the scalability problems of MSA parallel implementations while maintaining the accuracy of the alignments. The BGT is a new guide tree construction heuristic that consists of modifying the tree generation method to take into account not only the similarity between sequences, but also the balancing features. The 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. The BGT is not only able to improve the performance of T-Coffee but also does so without a lost of quality in the resulting alignment.

- The Optimized Library Method (OLM), which consists of an optimization method for the T-Coffee library to reduce the memory and CPU time requirements. This optimization was defined in two steps. The first one identified the information 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. 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.

- The Multiple Tree Alignment (MTA), which is 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 the default algorithm of T-Coffee and finally evaluating the resulting alignments to select the best one as the final result. The results show that MTA are able to recover and exceed the quality lost by the performance improvements.

The results obtained by applying the previous methods demonstrated that it is possible to improve the scalability of the consistency-based multiple-sequence aligners in both the execution time and also the number of sequences to be processed. In order to allow these improvements, it is necessary to use a High-Performance Computing infrastructure because of the greater computing and memory resources needed.

Our future work will be focused on aligning large-scale sequences (hundred of thousands). With the increasing performance of sequencing hardware, the need to align the genome of hundreds or thousands of individuals is not so far off. 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. The knowledge provided by the present paper will allow us to continue with this future work.

Acknowledgments: This work was supported by the Government of Spain TIN2011-28689-C02-02, TIN2010-12011-E, Consolider CSD2007-00050 and the CUR of GENCAT. Cedric Notredame is funded by the Plan Nacional BFU2008-00419 and the European Commission FP7, LEISHDRUG project (no 223414) and The Quantomics project (KBBE-2A-222664).

References

[1] S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, Oct. 1990.

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

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

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

[5] 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.

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

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

[8] 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.

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

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

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

[12] 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.

[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.

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

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

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

[17] D. Sankoff and J. H. Nadeau. *Comparative genomics*. Springer, 2000.

[18] F. Sievers, A. Wilm, D. Dineen, T. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Soding, J. Thompson, and D. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7(1), Oct. 2011.

[19] E. Sonnhamer, S. Eddy, and R. Durbin. Pfam: a comprehensive database of protein domain families based on seed alignments. *Proteins*, 28(3):405–420, '97.

[20] J. Thompson, D. Higgins, and T. 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, 1994.

[21] P. 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.

[22] C. Trapnell and M. Schatz. Optimizing data intensive gpgpu computations for dna sequence alignment. *Parallel Computing*, 35(8-9):429–440, 2009.

[23] 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.

[24] J. Zola, X. Yang, A. Rospondek, and S. Aluru. Parallel-tcoffee: A parallel multiple sequence aligner. In G. Chaudhry and S.-Y. Lee, editors, *ISCA PDCS*, pages 248–253. ISCA, 2007.