International Journal of Computer Applications (0975 - 8887)
Volume 182 - No.12, August 2018

A Survey of the State-of-the-Art Parallel Multiple Sequence Alignment Algorithms on Multicore Systems

Sara Shehab, Sameh Abdulah, Arabi E. Keshk

Department of Computer Science,
Faculty of Computers and Information,
Menofia University, Egypt

ABSTRACT

Evolutionary modeling applications are the best way to provide full information to support in-depth understanding of evaluation of organisms. These applications mainly depend on identifying the evolutionary history of existing organisms and understanding the relations between them, which is possible through the deep analysis of their biological sequences. Multiple Sequence Alignment (MSA) is considered an important tool in such applications, where it gives an accurate representation of the relations between different biological sequences. In literature, many efforts have been put into presenting a new MSA algorithm or even improving existing ones. However, little efforts on optimizing parallel MSA algorithms have been done. Nowadays, large datasets become a reality, and big data become a primary challenge in various fields, which should be also a new milestone for new bioinformatics algorithms.
This survey presents four different parallel MSA algorithms, *T-Coffee*, *MAFFT*, *MSAProbs*, and *M2Align*. We provide a detailed discussion of each algorithm including its strengths, weaknesses, and implementation details and the effectiveness of its parallel implementation compared to the other algorithms, taking into account the MSA accuracy on two different datasets, *BAliBASE* and *OXBench*.

Keywords

Bioinformatics; Multiple Sequence Alignment; Parallel Processing; Multicore Systems.

1. INTRODUCTION

For several years, studying the evolution of organisms plays an important role to understand their life, development, and physical structure. This kind of study requires an in-depth analysis of existing biological data to gain more information about different organisms. Thus, bioinformatics science is arisen as a new computation field to define a set of algorithms, methods, and techniques for understanding biological data [34]. Biological data is usually massive and requires developing and applying computationally intensive techniques.
One of the most challenging problem in bioinformatics is to extract the evolutionary relationship between different organisms. Assuming a set of biological sequences protein, DNA, or RNA sequences, the relation between two or more sequences can be shown as a sequence alignment process. Such a process is a fundamental tool in several applications, molecular function prediction, intermolecular interactions, residue selection, and phylogenetic analysis.
In this study, we concentrate on the Multiple Sequence Alignment (MSA) problem. In the literature, different sequential algorithms have been proposed to solve such a problem based on different methods. For example, progressive methods (ex., T-Coffee [24], Clustal), Multiple Alignment using Fast Fourier Transform (MAFFT) [17], Iterative methods (ex., MUSCLE [10]), Consensus methods (ex., M-COFFEE [32]), Hidden Markov models (i.e., HMMER [13]), and the intuitive methods (ex. PoMSA [27]). With the rapid growth of sequence databases, which now contains enough representatives of larger protein families to exceed the capacity of most current programs, the complexity of existing sequential algorithms is increased even in the case of two sequences alignment. For example, computations of current homologous sequence datasets could take several days.
In fact, the best methods sometimes fail to deal with these complexities efficiently and obtain biologically accurate alignments at the same time. The present studies overcome these obstacles by using two main approaches. The first is the vectorization, where all matrices are compensated by vectors, which in turn reduces the memory requirement and speed up execution without affecting the accuracy. The second approach is parallelism, the widespread programming method nowadays that allows multiple independent processes which share the same resources, to be executed concurrently at less time.
This paper concentrates on the parallel approach by studying four different shared-memory parallel implementations of the current state-of-the-art MSA algorithms, i.e., *T-Coffee* [8, 24], *MAFFT* [17, 18], *MSAProbs* [21], and *M2Align*[33] to show the strengths and weaknesses of each implementation. The importance of such a study is to illuminate the road for researchers to improve the existing implementation of these algorithms and provide new parallel solutions which became necessary with the sizable biological data we have today.

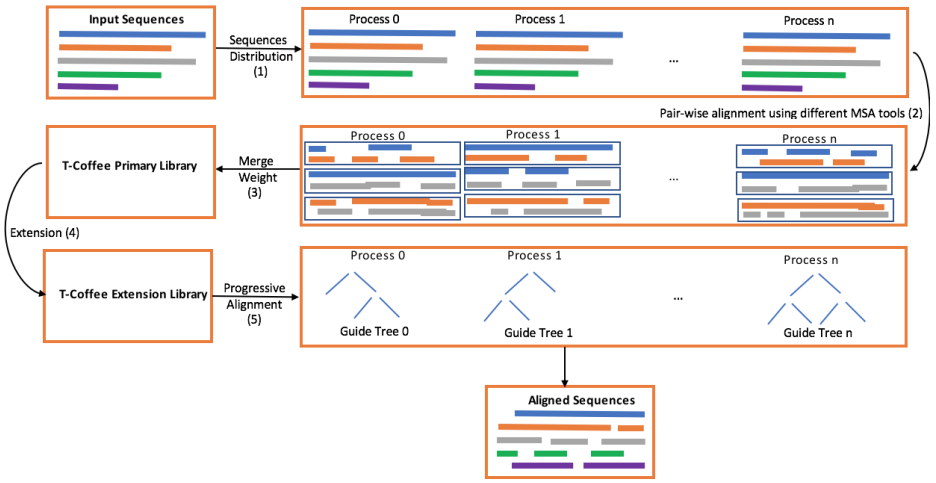


Fig. 1: Parallel T-coffee algorithm.

2. PARALLEL ALIGNMENT ALGORITHMS OVERVIEW

Although dynamic programming is the most optimal MSA solution according to the accuracy levels it can obtain, alignment several numbers of sequences is prohibitive because of the high computation requirement of such a solution [21, 23, 27]. Therefore, in literature, many heuristics provide a more practical solution with less computation complexity, such as progressive alignment [12], iterative alighnment [2], and alignment based on the profile Hidden Markov Models (HMM) [9]. Parallel algorithms also suffer from the memory bound restrictions that are provided by each algorithm. Thus, existing parallel implementations on shared memory mostly use MSA progressive alignment strategy to avoid memory overflow problems.
Here, a deep overview of four different parallel algorithms is given, *T-Coffee*, *MAFFT*, *MSAProbs*, and *M2Align*. The overview will cover both the sequential and parallel implementations for each algorithm.

2.1 T-Coffee Algorithm

Tree-based Consistency Objective Function for Alignment Evaluation algorithm (*T-Coffee*), is a progressive MSA algorithm, which optimizes the original progressive alignment in *Clustalw* algorithm [29]. The optimization involves using pre-aligned pair-wise sequences output (i.e., *T-Coffee* initial sources) from two or more pre-selected algorithms to improve the overall alignment operation. Even in the case of inconsistent pair-wise alignments in different sources, *T-Coffee* is able to differentiate the alignments of the same residue pairs by associating them with different weights.
The earliest implementation of *T-Coffee* algorithm has been proposed by [24, 29]. The proposed implementation has used two different pairwise alignment algorithms to initially aligned the given sequences, *ClustalW*, i.e., global pairwise aligner, and *lalign*, i.e., local pairwise aligner. Both algorithms generate a separate source of pairwise aligned sequences. *T-Coffee* assigns a separate weight to each aligned residue pair, which represents the correctness of the alignment before merging different sources as one primary library. During the merging step, *T-Coffee* can find the same residue pair with different alignment coming from different sources, in this case, each distinguish alignment should be represented as a separate entry in the *T-Coffee* primary library. However, if the same residue pair appears more than once in different sources with the same alignments, it should be added once to the primary library with a weight equals to the sum of the two weights.
The *T-Coffee* primary library can be used directly to perform MSA for the given sequences, however, more optimization can improve the whole alignment operation. For instance, another extension step can be applied to the primary library by adding another weight score that shows how a residue pair align is consistent with other sequences. In this case, the most consistent residue alignments are selected to be used in the MSA alignment step. The extended library now contains all the information that can improve the overall MSA alignment operation by using a traditional progressive alignment process.
In the progressive alignment technique, pairwise alignment library should be built first. In the case of the *T-Coffee* algorithm, the extension library can be used directly. *T-Coffee* depends on the dynamic programming to apply the progressive alignment strategy. The *T-Coffee* algorithm generates a distance matrix based on the extension library. The distance matrix is used to generate the guide tree that helps in generating the final list of aligned sequences.
In this study, we concentrate on the parallel implementations. In [8], a parallel implementation of the *T-Coffee* algorithm has been presented. The proposed parallel implementation targets four main parts of the algorithm, template selection, library computation, library extension, and progressive alignment. Template selection is a new feature added to most advanced modes of T-Coffee (i.e., R-Coffee, 3D-Coffee [1], and PSI-Coffee [4]) where input sequences are associated with structural templates [8]. In this study, we concentrate on the last three parts which are shown in the *T-Coffee* base algorithm.
The library computation step is related to the use of different pairwise alignment algorithms to provide a set of initial libraries to the *T-Coffee* algorithm. This step can be parallelized by dividing the alignment task into several tasks that are equal to the number of available cores. The master process merges the outputs coming from different cores into a single library.
The extension library generation step can also be parallelized by distributing the residue pairs to the available cores so that extended

weights can be added simultaneously. Moreover, the progressive alignment step is parallelized by processing the guide tree with different processes. Each process deals with only one node independently to generate the final output. Figure 1 gives an overview of parallel implementation of *T-Coffee* algorithm.

2.2 MAFFT Algorithm

Multiple sequence Alignment based on Fast Fourier Transform (*MAFFT*), is an MSA algorithm which depends on the fast Fourier transform algorithm to fast discover matched parts of a given set of sequences. *MAFFT* has three heuristic running modes, progressive method (FFT-NS-2), the iterative refinement method (FFT-NS-i, L-INS-i, E-INS-i, and G-INS-i), and the structural alignment method for RNA (Q-INS and X-INS-i) [17]. Each mode is suitable for a certain number of sequences and the user priority (i.e., fast or accurate computation).
The progressive method in *MAFFT* is a straightforward method of generating the distance matrix and the guide tree to better align the given set of sequences. This option usually used in the case of a very large number of sequences and fast computation priority from the user. In the iterative refinement method, accurate alignment is considered a priority and several alignment iterations are required. Further, structural alignment method is more suitable for RNA alignment proteins with low sequence similarity.
Sequential *MAFFT* has three main steps, all-to-all comparison, progressive alignment, and iterative refinement. The all-to-all comparison can be considered as a part of the progressive alignment step where a set of pairwise comparisons are conducted to generate the initial distance matrix and the guide tree that are used by the progressive alignment step. Progressive alignment is the main MSA process while iterative refinement step is used to improve the accuracy of the final result.
In [18] a parallel *MAFFT* algorithm has been proposed. The given implementation is based on POSIX Threads library and targets the three main steps of the *MAFFT* algorithm. All-to-all comparison has been parallelized by distributing the pairwise alignment to the available physical cores. In progressive alignment, parsing the guide tree is parallelized every level, i.e., each tree parent depends on its child which prevents parallelization over the whole tree nodes. Finally, if iterative refinement is used, two different approaches have been proposed, *best-first approach* and *hill-climbing approach*, where both of them aim at dividing each alignment into two sub-alignments and the two sub-alignments are realigned. The realignment step is performed according to the tree dependent iterative strategy proposed in [14].
In the *best-first approach*, The realignments are performed for all possible alignments on the iterative tree. The alignment with the highest objective score is selected for another iteration. Once no high score alignment is found, the algorithm terminates. In the *hill-climbing* approach, each process has its local realignments of the original alignment and the better score is replaced the original alignment is kept locally by each process. In this case, many realignments of the same alignment are used in the next iteration [18]. Figure 1 gives an overview of parallel *MAFFT* algorithm.

2.3 MSAProbs

MSAProbs is a progressive algorithm which provides an effective solution to perform MSA by combining both Hidden Markov Models (HMM) technique with partition functions to calculate posterior alignment probabilities. MSAProbs is able to provide a higher alignment accuracy through two new techniques, weighted probabilistic consistency transformation and weighted profile-profile alignment [21].
Sequential MSAProbs consists of five main stages, generate pairwise posterior probability matrices using a pair-HMM and a partition function, generate a pairwise distance matrix using the set of posterior probability matrices, constructing a guide tree from the pairwise distance matrix to estimate different sequences weights, performing a weighted probabilistic consistency transformation of all pairwise posterior probability matrices, and apply a progressive alignment on the *guide tree* using the transformed posterior probability matrices. In-depth details about each step are shown in [21].
In this paper, we concentrate more on the parallel implementation of MSAProbs which is available under MSAProbs v0.9.7 package [20]. Among the five MSAProbs' steps, generating the pairwise posterior probability matrices, i.e., step 2, and weighted probabilistic consistency transformation, i.e., step 4 , are the most time consuming part with time complexity of $\mathcal{O}(N^2L^2)$ and $\mathcal{O}(N^2L^3)$, respectively. N represents the number of sequences and L represents the average length of sequences.
Current implementation of parallel MSAProbs is mostly bases on parallizing these two most consuming steps in the algorithm using OpenMP, i.e., a compiler-directive-based API for shared-memory parallelism [6]. Figure 3 shows an overview of parallel MSA-Probs algorithm.

2.4 M2Align

Multi-objective metaheuristics methods is another approach to deal with MSA problems. In such methods, a non-exact stochastic optimization function is used to highly optimized two or more objectives concurrently. A Matlab-based tool called MO-SAStrE is an example of applying such an approach to the MSA problem [25]. This tool depends on genetic algorithms through a set of mutation and crossover operations to the expected alignment results to find the best alignment solution for a set of given sequences. NSGA-II genetic algorithm is the backbone of MO-SAStrE tool [7].
The *M2align* algorithm is a parallel multiple sequence alignment algorithm based on MO-SAStrE. Besides exploiting the multi-core architecture of current computing devices to parallelize the MO-SAStrE algorithm, it optimizes the functionality of the algorithm by using a better storage mechanism to store gaps information during the optimization phase.
The *M2align* algorithm is working as follows. A set of predetermined MSA algorithms is used to generate N alignment solutions which are used as the initial population in the optimization phase. Another set of solutions can be added to the population through applying genetic crossover operators (i.e., Single-Point Crossover) to each pair of the initial N solutions. later, a set of N solutions is selected based on its dominance ranking [33]. An initial evaluation step is applied to assign a dominance rank for each set of sequences. Figure 4 illustrates the whole functionality of *M2align* algorithm over a given set of sequences.
The evaluation step in MO-SAStrE is a sequential step which takes a longer time to complete compared to the other algorithm steps. Thus, M2align algorithm includes a parallel implementation of this step to speedup the alignment process through multi-core systems. In this case, parallelization has been exploited by evaluating solutions concurrently at the same time using differently available cores.

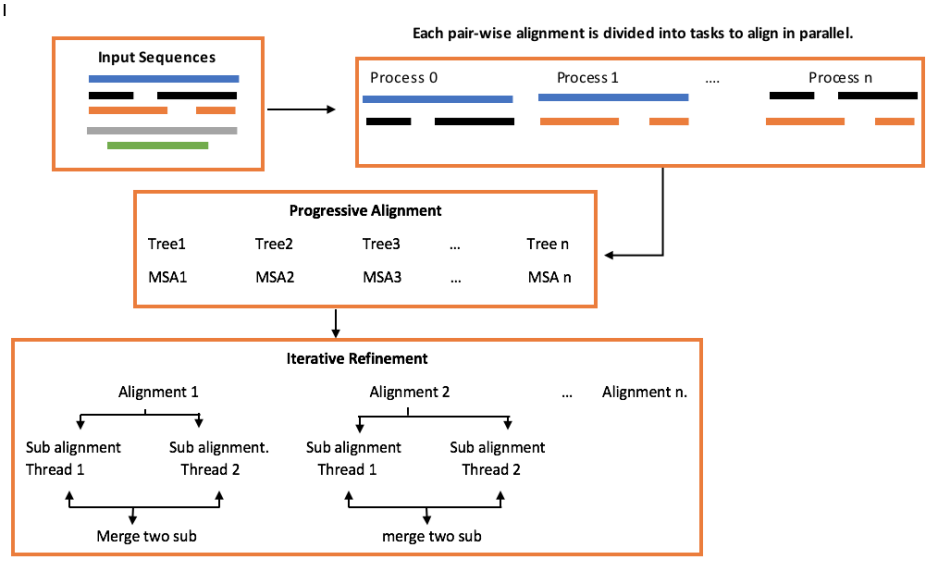


Fig. 2: Parallel MAFFT algorithm.

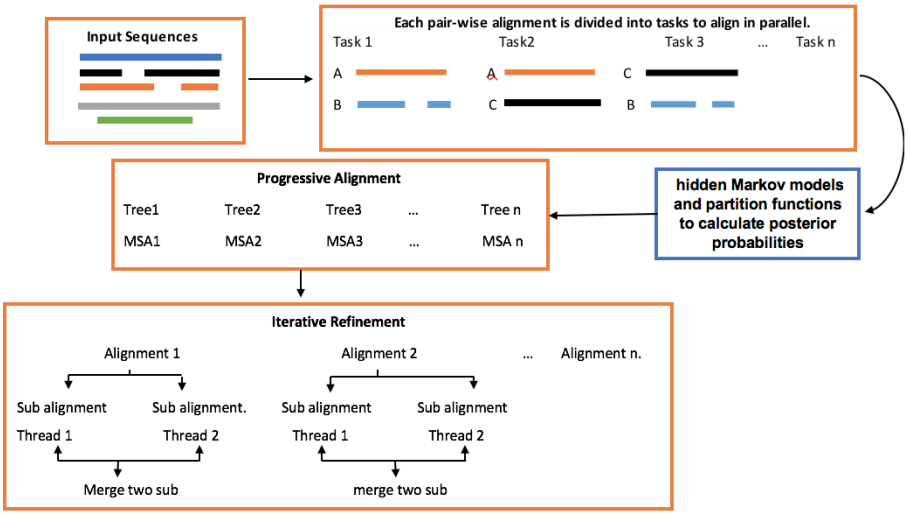


Fig. 3: Parallel MSA-Probs algorithm.

3. PERFORMANCE EVALUATION

In this section, we present a detailed performance evaluation of the four parallel MSA algorithms: *T-Coffee*, *T-MAFFT*, *MSAProbs*, and *M2Align*. The evaluation involves execution time and the obtained accuracy of these algorithms using a different number of cores on two different datasets.

3.1 Setup

3.1.1 Experimental Testbed. We evaluate the performance of the target software packages on a dual-socket 14-core Intel Broadwell E5-2680 V4 processor running at 2.4 GHz. The software packages are compiled with gcc v4.8 on Ubuntu 16.04.3 LTS.

Two datasets have been used for the evaluation, BAliBASE [31], and OXBench [26]. To validate the accuracy of different MSA algorithms, we use *Bench* [3], which provides different benchmarks for several proteins datasets.

3.2 Accuracy Assessment

We uses Three different metrics to evaluate the alignment accuracy of the given MSA algorithms,

(1) *Q/TC score*: Q score represents the number of correctly aligned residue pairs divided by the number of residue pairs in the reference alignment, i.e., Bench. TC, i.e., total column score, represents the number of correctly aligned columns

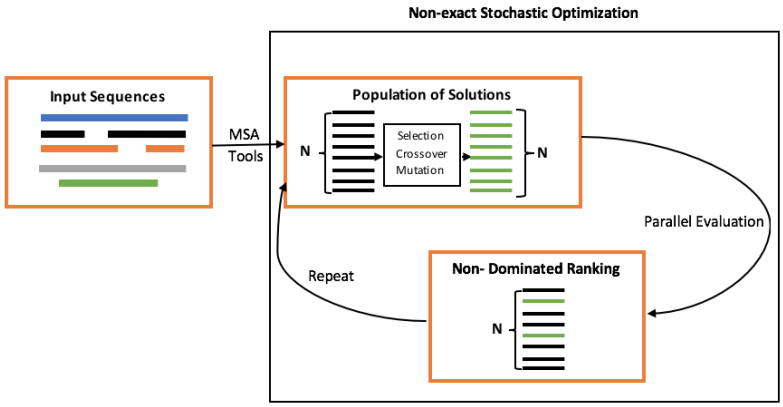


Fig. 4: Parallel M2align algorithm.

divided by the number of columns in the reference alignment [10].

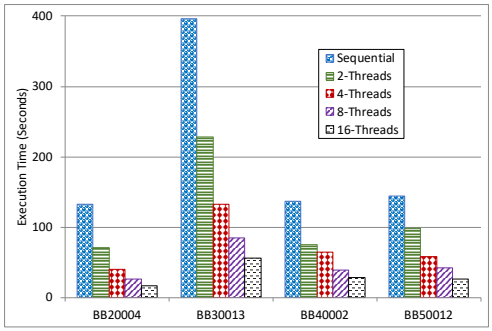
(2) *The Cline's score (CS) or shift score*, this score mainly represents a distance-based scoring. It is negatively impacted by both over-alignment where non-homologous regions are aligned or under-alignment where homologous regions are not aligned improbably [5]. CS is considered as the best accuracy metric compared to existing scoring metrics [11, 22].

(3) *The modeler's score* represents the number of correctly aligned pairs in the aligned file divided by the number of pairs in the reference file [22].

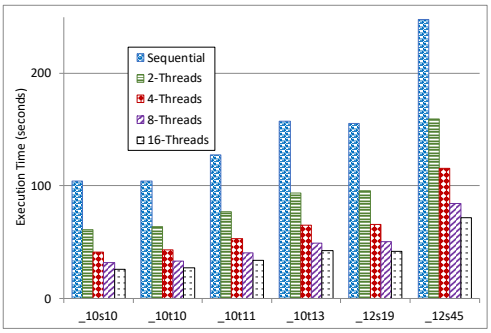
3.3 Performance Comparison

Most of the existing MSA studies concentrate on the alignment accuracy level that can be obtained by the given MSA algorithm. However, in this study, we are mostly focusing on the parallel execution time which should be associated with an acceptable scalability trait. Scalability is considered an important factor that shows the ability of a software to handle a growing amount of work with a larger number of processes.
The following experiments show the performance of our target parallel MSA algorithms using a different number of threads (i.e., 2, 4, 8, and 16) besides the sequential implementation of each algorithm. We assume each core hosts only one running thread (i.e., hyper-threading is disabled). Our target Broadwell multi-core system has a total of 28 cores. However, due to the scalability issue of the existing software implementations, we perform our experiments with up to 16 threads.

3.3.1 Parallel T-Coffee Algorithm Performance. We evaluate *T-Coffee* implementation using 1, 2, 4, 8, and 16 threads. Figures 5(a) and 5(b) show the execution scalability using Balibase and OXbench datasets, respectively. The figures show linear scalability of T-Coffee with a larger number of threads. For example, in the case of BAliBASE dataset, the speedup from sequential execution to 16 threads execution are $6.8X$, $6.1X$, $3.9X$, $4.5X$ in $BB20004$, $BB30013$, $BB40002$, $BB50012$ files, respectively. The same observation can be drawn from Figure 5(b). The speedup in the case of OXbench datasets are $3X$, $2.8X$, $2.8X$, $2.7X$, $2.7X$, $2.4X$ in $\_10s10$, $\_10t10$, $\_10t11$, $\_10t13$, $\_12s19$, $\_12s45$ files, respectively.



(a) Parallel T- Coffee - Balibase Dataset.

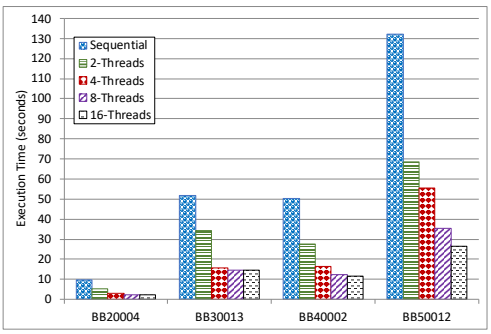


(b) Parallel T- Coffee - OXbench Dataset.

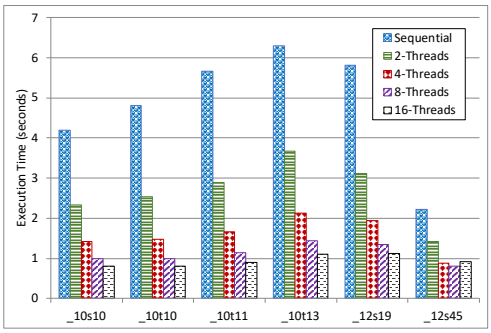
Fig. 5: Parallel T- Coffee total execution time with different number of threads.

3.3.2 Parallel MAFFT Algorithm Performance. The same set of experiments have been conducted using parallel *MAFFT*. Figure 6 shows the execution time using both Balibase and OXbench datasets.
MAFFT has three running options, i.e., progressive, iterative refinement, and structural alignment. According to the dataset size, only one option can be chosen. Here, we choose an *auto* option, which allows the algorithm to pick the best option for each dataset.
Figure 6(a) shows the performance on Balibase dataset. The gained speedup from sequential execution to 16 thread execution is $3.73X$,

$2.56X$, $3.32X$,and $4.02X$ in $BB20004$, $BB30013$, $BB40002$, $BB50012$ files, respectively. Figure 6(b) shows the execution time using OXbench dataset. The speedup from sequential to 16 threads execution is $4.22X$, $5.01X$, $5.34X$, $4.68X$, $4.17X$, and $1.44X$ in _$10s10$, _$10t10$, _$10t11$, _$10t13$, _$12s19$, _$12s45$ files, respectively. The parallel *MAFFT* scalability can be shown by both figures.



(a) Parallel MAFFT - Balibase dataset.

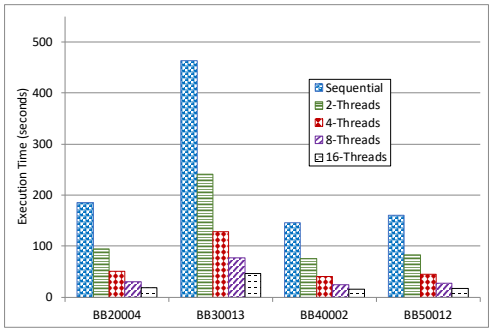


(b) Parallel MAFFT - OXbench dataset.

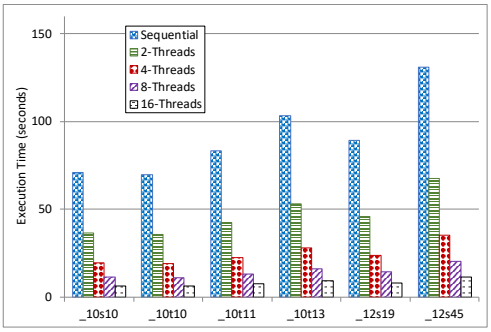
Fig. 6: Parallel MAFFT total execution time with different number of Threads.

3.3.3 Parallel MSAProbs Algorithm Performance. The *MSAProbs* algorithm also shows a high scalability with a different number of threads on the given set of files. Figure 7 shows the performance on both Balibase data files, i.e., Figure 7(a), and OXbench data files, i.e., Figure 7(b).
Figure 7(a) shows that *MSAProbs* is able to achieve around $9X$ speedup using 16 threads compared to sequential implementation across different files from the Balibase dataset. Further, Figure 7(b) shows an average speedup of $11X$ across the *OXbench* dataset's files.

3.3.4 M2Align. Figure 8 shows the parallel execution performance of *M2Align* algorithm on Balibase dataset. The speedup from sequential execution to 16 threads executions are $7.85X$, $9.41X$, $7.50X$, and $10.55X$ in $BB20004$, $BB30013$, $BB40002$, $BB50012$ files, respectively. As shown,the parallel *M2Align* satisfies a higher speedup with larger number of threads compared to both *T-Coffee*, *MAFFT*, and *MSAprobs* algorithms.
The same case can also be shown by Figure 8(b), where MSAProbs is used on OXbench dataset. The speedup is $7.43X$, $4.22X$m $3.91X$, $7.79X$, $0.82X$, and 0.91 in _$10s10$, _$10t10$, _$10t11$, _$10t13$, _$12s19$, _$12s45$ files, respectively.



(a) Balibase dataset with parallel MSAProbs.



(b) OXbench dataset with parallel MSAProbs.

Fig. 7: Parallel MSAProbs total execution time with different number of threads.

An important observation can be shown in Figure 8(b). With less sequential execution time, the scalability becomes worse and the speedup decreases, i.e., _$12s19$, _$12s45$ files.

3.4 Accuracy Comparison

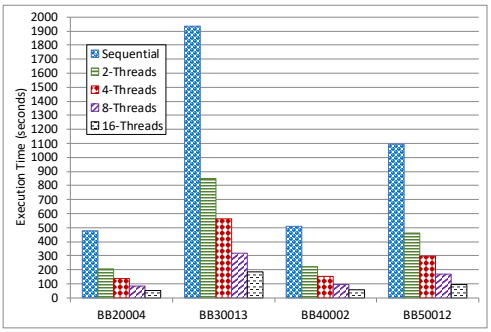
Evaluating the efficiency of an MSA algorithm is related to the final alignment score it able to obtain. As mentioned before, in this study we use three different scoring metrics, Q/TC, $cline$, $modeler$. All the scoring metrics use $Bench$ benchmark to estimate the accuracy score of different algorithms.
Table 1 shows the Q/TC accuracy using our four target algorithms on nine different files from BAliBASE and OXBench datasets. The average scoring values show that *T-Coffee* algorithm is the most efficient algorithm. *MSAProbs* algorithm comes in the second order followed by the *MAFFT* algorithm and finally the *M2align* algorithm.
Table 2 shows the average accuracy scores of each algorithm using the same set of files. The tables show that MSAProbs outperforms all other algorithms and T-Coffee algorithm is coming second followed by the MAFFT and the M2align algorithms.
Table 3 also shows the average $modeler$ score of the algorithms over the same set of files. The scores show that the *MSAProbs* algorithm satisfies the highest average score through all files. The *T-Coffee* algorithm satisfies the second highest score followed by the *MAFFT* and the *M2align* algorithms.

3.5 Speedup Comparison

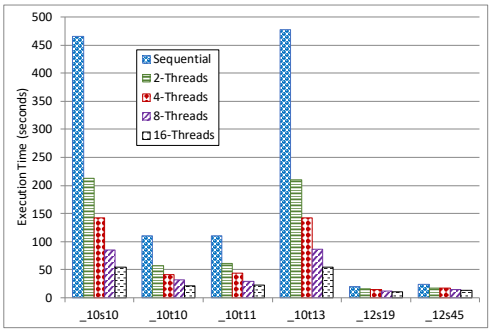
Increasing the number of workers in parallel execution usually leads to improve the performance of the underlying application. In

Table 1. : Accuracy comparison using Q/TC score.

Dataset	Q/TC			
	MAFFT	T-Coffee	MSAProbs	M2align
BB20004	0.973/0.787	0.984/0.787	0.985/0.792	0.981/0.733
BB40002	0.539/0	0.99/0.737	0.908/0	0.574/0
BB50012	0.866/0.429	0.896/0.49	0.89/0.51	0.877/0.327
_10s10	0.926/0.667	0.926/0.667	0.926/0.667	0.926/0.667
_10t10	0.833/0.741	0.87/0.815	0.87/0.815	0.735/0.593
_10t11	0.603/0.429	0.702/0.619	0.603/0.429	0.502/0.238
_10t13	0.917/0.667	0.917/0.667	0.917/0.667	0.733/0
_12s19	1	1	1	1
_12s45	1	1	1	1
Average	0.8507/0.6355	0.9205/0.7535	0.8998/0.6533	0.8142/0.506



(a) Balibase dataset with parallel M2Align.



(b) OXbench dataset with parallel M2Align.

Fig. 8: M2Align total execution time with different number of threads.

Table 2. : Accuracy comparison using Cline score.

Dataset	Cline			
	MAFFT	T-Coffee	MSAProbs	M2align
BB20004	0.572	0.576	0.578	0.576
BB40002	0.1	0.25	0.317	0.113
BB50012	0.38	0.399	0.406	0.378
_10s10	0.11	0.11	0.11	0.109
_10t10	0.281	0.291	0.292	0.262
_10t11	0.178	0.201	0.185	0.164
_10t13	0.0386	0.0386	0.0387	0.0332
_12s19	0.257	0.257	0.257	0.257
_12s45	0.434	0.434	0.436	0.434
Average	0.2611	0.2840	0.2910	0.258

Table 3. : Accuracy comparison using Modeler score.

Dataset	Modeler			
	MAFFT	T-Coffee	MSAProbs	M2align
BB20004	0.4	0.402	0.404	0.402
BB40002	0.0388	0.0752	0.149	0.0325
BB50012	0.219	0.23	0.241	0.206
_10s10	0.0562	0.0564	0.0563	0.0582
_10t10	0.161	0.168	0.168	0.15
_10t11	0.09	0.104	0.0908	0.0763
_10t13	0.0193	0.0193	0.0194	0.0158
_12s19	0.147	0.147	0.147	0.147
_12s45	0.278	0.278	0.278	0.278
Average	0.1565	0.1644	0.1726	0.1517

this subsection, we evaluate the gained speedup of increasing the number of cores. Because *M2Align* algorithm gives the lowest execution time through all given data files, we use it as a benchmark of the performance of the other algorithms. The main goal of this experiment is to show if a particular algorithm speedup increases with a larger number of cores.

Figure 9 illustrates the average gained speedup using a different number of threads compared to the sequential version of each algorithm on different data files.

As shown, both *MSAProbs* and *M2Align* algorithms can respectively gain up to 1.93X and 2.27X speedup using 2-threads, 3.61X and 3.41X speedup using 4-threads, 6.07X and 5.79X speedup using 8-threads, and 10.04X and 9.46X speedup using 16-threads.

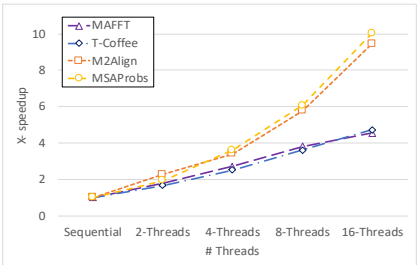


Fig. 9: *T-Coffee*, *MAFFT*, *MSAProbs*, and *M2Align* average speedup comparison using different number of threads on different data files.

Both algorithms show super-linear scalability with a larger number of threads.
Further, *T-Coffee* and *MAFFT* algorithms can respectively gain up to 1.67X and 1.80X speedup using 2-threads, 2.52X and 2.73X speedup using 4-threads, 3.62X and 3.82X speedup using 8-threads, and 4.72X and 4.56X speedup using 16-threads. Both algorithms show linear scalability with a larger number of threads.
The figures show the strong scalability of both *MSAProbs* and *M2Align* compared to both *T-Coffee* and *MAFFT* algorithms.

4. CONCLUSION

The literature reveals a lack of providing a detailed comparison between existing parallel MSA algorithms to show their strengths and weaknesses. Thus, this study mainly aims at studying the state-of-the-art multicore-based MSA algorithms from two different perspectives, performance, and accuracy. Four different algorithms have been picked up for this study, *T-Coffee*, *MAFFT*, *MSAProbs*, and *M2Align*. A detailed explanation of each algorithm has been given associated with its parallel implementation on multicore systems. Several software packages have been also introduced to show the effectiveness of the given algorithms on parallel platforms.
Further, this work shows an intensive performance evaluation on Intel Broadwell 14-core system using up to 16 threads. The performance comparison shows a linear scalability of the four algorithms using a different number of threads. Moreover, A comprehensive qualitative assessment of the accuracy score of each algorithm has been conducted using three different metrics, *Q/TC*, *cline*, and *modeler*. The accuracy comparison shows better results using both T-Coffee and *MSAProbs* comparing to *MAFFT* and *M2Align* algorithms.
In the future, we aim at extending this survey to include both distributed and GPU based parallel MSA algorithms by covering the alignment of a large number of sequences on large-scale systems. The study should cover the impact of the availability of large-scale hardware resources on the performance of existing MSA solutions.

References

[1] Fabrice Armougom, Sebastien Moretti, Olivier Poirot, Stephane Audic, Pierre Dumas, Basile Schaeli, Vladimir Keduas, and Cedric Notredame. Expresso: automatic incorporation of structural information in multiple sequence alignments using 3d-coffee. *Nucleic acids research*, 34(suppl_2):W604–W608, 2006.

[2] Geoffrey J Barton and Michael JE Sternberg. A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *Journal of molecular biology*, 198(2):327–337, 1987.

[3] Bench. Multiple Sequence Alignment (MSA) benchmark, 1999.

[4] Jia-Ming Chang, Paolo Di Tommaso, Jean-François Taly, and Cedric Notredame. Accurate multiple sequence alignment of transmembrane proteins with psi-coffee. *BMC bioinformatics*, 13(4):S1, 2012.

[5] Melissa Cline, Richard Hughey, and Kevin Karplus. Predicting reliable regions in protein sequence alignments. *Bioinformatics*, 18(2):306–314, 2002.

[6] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[8] Paolo Di Tommaso, Miquel Orobitg, Fernando Guirado, Fernando Cores, Toni Espinosa, and Cedric Notredame. Cloudcoffee: 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.

[9] Sean R. Eddy. Profile hidden markov models. *Bioinformatics (Oxford, England)*, 14(9):755–763, 1998.

[10] Robert C Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[11] Robert C Edgar and Kimmen Sjölander. A comparison of scoring functions for protein sequence profile alignment. *Bioinformatics*, 20(8):1301–1308, 2004.

[12] Da-Fei Feng and Russell F Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of molecular evolution*, 25(4):351–360, 1987.

[13] Robert D Finn, Jody Clements, and Sean R Eddy. Hmmer web server: interactive sequence similarity searching. *Nucleic acids research*, 39(suppl_2):W29–W37, 2011.

[14] Osamu Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Bioinformatics*, 9(3):361–370, 1993.

[15] X Huang and W Miller. Lalign-find the best local alignments between two sequences. *Adv. Appl. Math*, 12:373, 1991.

[16] Kazutaka Katoh, Kazuharu Misawa, Kei-ichi Kuma, and Takashi Miyata. Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. *Nucleic acids research*, 30(14):3059–3066, 2002.

[17] Kazutaka Katoh and Daron M Standley. Mafft multiple sequence alignment software version 7: improvements in performance and usability. *Molecular biology and evolution*, 30(4):772–780, 2013.

[18] Kazutaka Katoh and Hiroyuki Toh. Parallelization of the mafft multiple sequence alignment program. *Bioinformatics*, 26(15):1899–1900, 2010.

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

[20] Yongchao liu. MSAProbs - Parallel and accurate multiple sequence alignment. `http://msaprobs.sourceforge.net/homepage.htm`, 2018. [Online; accessed 25-May-2018].

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

[22] Dimitrios P Lyras and Dirk Metzler. Reformalign: improved multiple sequence alignments using a profile-based meta-alignment approach. *BMC bioinformatics*, 15(1):265, 2014.

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

[24] Cédric Notredame, Desmond G Higgins, and Jaap Heringa. T-coffee: a novel method for fast and accurate multiple sequence alignment1. *Journal of molecular biology*, 302(1):205–217, 2000.

[25] Francisco M Ortuno, Olga Valenzuela, Fernando Rojas, Hector Pomares, Javier P Florido, Jose M Urquiza, and Ignacio Rojas. Optimizing multiple sequence alignments using a genetic algorithm based on three objectives: structural information, non-gaps percentage and totally conserved columns. *Bioinformatics*, 29(17):2112–2121, 2013.

[26] GPS Raghava, Stephen MJ Searle, Patrick C Audley, Jonathan D Barber, and Geoffrey J Barton. Oxbench: a benchmark for evaluation of protein multiple sequence alignment accuracy. *BMC bioinformatics*, 4(1):47, 2003.

[27] Sara Shehab, Sameh Shohdy, and Arabi E Keshk. Pomsa: An efficient and precise position-based multiple sequence alignment technique. *Journal of Advanced Research in Computing and Applications*, 9:14–20, 2017.

[28] JD Thomopson, Desmond G Higgins, and Toby J Gibson. Clustalw. *Nucleic Acids Res*, 22:4673–4680, 1994.

[29] Julie D Thompson, Toby Gibson, Des G Higgins, et al. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, pages 2–3, 2002.

[30] Julie D Thompson, Toby J Gibson, and Des G Higgins. Multiple sequence alignment using clustalw and clustalx. *Current protocols in bioinformatics*, (1):2–3, 2003.

[31] Julie D. Thompson, Frédéric Plewniak, and Olivier Poch. Balibase: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics (Oxford, England)*, 15(1):87–88, 1999.

[32] Iain M Wallace, Orla O'sullivan, Desmond G Higgins, and Cedric Notredame. M-coffee: combining multiple sequence alignment methods with t-coffee. *Nucleic acids research*, 34(6):1692–1699, 2006.

[33] Cristian Zambrano-Vega, Antonio J Nebro, José García-Nieto, and Jose F Aldana-Montes. M2align: parallel multiple sequence alignment with a multi-objective metaheuristic. *Bioinformatics*, 33(19):3011–3017, 2017.

[34] Albert Y Zomaya. *Parallel computing for bioinformatics and computational biology: models, enabling technologies, and case studies*, volume 55. John Wiley & Sons, 2006.