


A Greedy Clustering Algorithm for Multiple Sequence Alignment

Rabah Lebsir, Computer Science Department, Faculty NTIC, University Constantine 2, MISC Laboratory, University Constantine 2, Algeria & University of Guelma, Algeria

 <https://orcid.org/0000-0002-4418-1814>

Abdesslem Layeb, Computer Science Department, Faculty NTIC, University Constantine 2, Algeria

Tahi Fariza, University of Évry Val d'Essonne, France & *Université Paris-Saclay, France*

ABSTRACT

This paper presents a strategy to tackle the multiple sequence alignment (MSA) problem, which is one of the most important tasks in the biological sequence analysis. Its role is to align the sequences in their entirety to derive relationships and common characteristics between a set of protein or nucleotide sequences. The MSA problem was proved to be an NP-Hard problem. The proposed strategy incorporates a new idea based on the well-known divide-and-conquer paradigm. This paper presents a novel method of clustering sequences as a preliminary step to improve the final alignment; this decomposition can be used as an optimization procedure with any MSA aligner to explore promising alignments of the search space. In their solution, the authors proposed to align the clusters in a parallel and distributed way in order to benefit from parallel architectures. The strategy was tested using classical benchmarks like BALiBASE, Sabre, Prefab4, and Oxm, and the experimental results show that it gives good results by comparing to the other aligners.

KEYWORDS

Biological Sequences, Clustering, Local Search, Multi-Core, Multiple Sequence Alignment, Parallel Algorithm

INTRODUCTION

The multiple sequence alignment (MSA) consists to align more than two biological sequences like DNA or protein to bring out similar or homologous regions. MSA plays an important task in Bioinformatics and it is widely used like in protein analysis, identification of functional sites in genomic sequences, structural prediction, etc. Unfortunately, finding an optimal MSA has been demonstrated NP-hard (Wang & Jiang, 1994). Indeed, MSA is an optimization problem, which exhibits a high time and space complexity. Accordingly, to solve this problem, several methods were proposed. They can be categorized into three classes (Notredame, 2002): exact methods, progressive methods and iterative methods.

The exact methods are based on a generalization of Needleman algorithm (Needleman & Wunsch, 1970) to align sequences simultaneously. Although the exact methods provide optimal solutions, their main disadvantage is their high complexity in time and space. Consequently, they quickly became impracticable in the case of high number and length of sequences. The second class contains methods

DOI: 10.4018/IJCINI.20211001.0a41

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

based on a progressive approach (Feng & Doolittle, 1987). The MSA is built up by combining pairwise alignments, starting by the pair with high similarity and succeeding to the most distantly related one. Progressive methods are widely used and are effective, simple, fast and give generally a good quality alignment. However, their main disadvantage is the local minima problem and they, therefore, may lead to poor solutions. The third class consists of iterative methods (Hirosawa, Totoki, Hoshida, & Ishikawa, 1995). The idea is to start by a preliminary alignment and then refine it iteratively through appropriate improvements called iterations. The process is repeated until the satisfaction of certain criteria. Due to the computationally intensive operation of MSA algorithms and large amount of available sequence data, MSA has been gaining importance again in recent years and many parallel MSA algorithms were developed, such as Clustal Omega (Sievers & Higgins, 2014).

In this paper, another idea to perform the multiple alignment problem is investigated. The objective of the proposal work is to increase the accuracy while avoiding wasting too much time. The proposed approach uses primarily a clustering strategy based on a local search to divide sequences into subsets, and then uses an alignment algorithm in each subset to build a multiple alignment for each cluster, and at the end, it uses the same algorithm to build an alignment for the consensus of the subsets to generate the MSA for all sequences.

Experiments on a wide range of biological data benchmarks have shown the effectiveness of the proposed technique and its ability to generate an alignment with good accuracy. Experiments show also the ability of the proposed technique to be used in the case of parallel or distributed architectures to further improve the processing time for large-scale biological data.

The paper is organized as follows. In Section 1, a formulation of the tackled problem is given. Section 2 presents state-of-the-art methods for resolving the MSA problem. Section 3 presents the techniques used to improve the speed of MSA. Section 4 is devoted to the presentation of our proposed algorithm. Section 5 performs experimentations for monitoring and evaluating the performance of the proposed work. Finally, Section 6 outlines the main conclusions.

MSA PROBLEM FORMULATION

In this section, the formulation of MSA problem is given to show its combinatorial nature. Let $S = \{s_1, s_2, s_3, \dots, s_n\}$ a set of n sequences with $n \geq 2$. Each sequence s_i is a string well-defined over an alphabet. The lengths of the sequences are not necessarily the same and in most of time are different. The MSA problem can be well-defined by identifying a pair (Ω, C) where Ω is the set of all potential alignments and C define a function $\Omega \rightarrow \mathbb{R}$ called the alignment score. Each possible alignment is seen as a set $S' = \{s'_1, s'_2, s'_3, \dots, s'_n\}$ satisfying the following criteria:

- Each sequence s'_i is an extension of s_i and is defined over the alphabet $\Delta' = \Delta \cup \{-\}$. The symbol $' - '$ denotes a gap. When gaps are deleted from s'_i , s'_i and s_i are identical.
- For all i, j , $\text{length}(s'_i) = \text{length}(s'_j)$.
- An alignment score of S' represented by $C(S')$ is defined as: $C(S') = \sum_i \sum_j \text{sim}(s'_i, s'_j)$ where $\text{Sim}(s'_i, s'_j)$ represents similarities between each pair of sequences s'_i and s'_j . The optimum value of C can be defined as: $C_{\text{best}} = \left\{ \max C(S') / S' \in \Omega \right\}$ and the optima set Ω_{best} as: $\Omega_{\text{best}} = \left\{ S' \in \Omega / C(S') = C_{\text{best}} \right\}$

Example: Let $S = \{s_1, s_2, s_3\}$ a set of 3 protein sequences, $s_1 = \text{AGMASGYD}$; $s_2 = \text{KASYD}$; and $s_3 = \text{AGASYD}$. A possible alignment can be presented as a set $S' = \{s'_1, s'_2, s'_3\}$ where: $s'_1 = \text{AGMASGYD}$; $s'_2 = \text{KA--S-YD}$ and $s'_3 = \text{AG-AS-YD}$. The MSA can be presented as the following:

```
A G M A S G Y D
K A - - S - Y D
A G - A S - Y D
```

The addressed task is clearly a combinatorial optimization problem. The method proposed here does not make explicit use of a score function to assess the quality of a whole alignment and therefore to guide the search in the space of alignments. Instead, it uses a domain decomposition strategy and a heuristic based on the distance between sequences that allows exploring the search space in a specific way.

STATE-OF-THE-ART METHODS FOR MSA

Since generating an optimal MSA is characterized by its high complexity in terms of time and space, many algorithms have been developed to find near optimal solutions, the MSA methods are generally divided in three main classes:

Exact Alignments

Exact algorithms were developed to align sequences simultaneously. They are able to generate an optimal alignment, but they can only support a small number of sequences. Thus, high memory requirement, high computational effort and limitation on the number of sequences are the main weakness of these methods.

Generally, the exact methods are a generalization of the Needleman-Wunsch dynamic programming algorithm for the multiple alignment of n sequences by using an n -dimensional score matrix. The value of each box in the matrix depends on its neighbors, so for N sequences of length L , the size of the matrix is $N \cdot L$. This approach is so greedy in terms of resources that it becomes impracticable for $N > 4$.

Progressive Alignments

Progressive methods are the most popular methods. Progressive alignment methods repetitively apply pairwise alignment algorithms to build a MSA. These methods are simple, fast and don't need a large memory. Given n sequences of length L , the runtime for progressive alignment is $O(nL^2)$ (Wang & Jiang, 1994). Basically, the main steps of these algorithms are the following:

1. Choose two sequences and align them.
2. Choose another one and align it with the consensus of sequences previously aligned.
3. Repeat step 2 until the alignment of all sequences.

However, the problem in these methods is to determine the order according to which the sequences are aligned. One of the more promising solutions is the use of the guide tree suggested by Feng and Doolittle (Feng & Doolittle, 1987). The alignment will be constructed following the guide tree containing the order. After the alignment of the two closest sequences, more sequences are added by aligning them with the existing alignment. Aligning partial alignments is also possible. Alignment along a tree does not necessarily yield an optimal alignment, even if the tree is perfect. For example, errors made in early stages are propagated to later stages and cannot be fixed. A variety of MSA

programs are created based on the Feng and Doolittle algorithm such as CLUSTAL (Thompson, Higgins, & Gibson, 1994). The construction of the similarity tree is the main difference between these programs; for example, CLUSTAL uses the Neighbor-Joining algorithm to generate the guide tree. Generally, the basic steps of Feng-Doolittle algorithm are the following:

1. Calculate all possible pairwise alignments scores between all sequences.
2. Convert the alignments scores into distances and construct the guide tree by using the distance matrix and a tree construction method.
3. A multiple alignment is created gradually starting by the closest sequences. The sequences are added one by one depending on the order given by the guide tree.

Progressive aligners can be global or local according to the pairwise alignment algorithm used. In the case of global progressive alignment such as CLUSTAL (Thompson et al., 1994), a global pairwise algorithm such as Needleman algorithm (Needleman & Wunsch, 1970) is used to align each unaligned sequence to the sequences previously aligned in the previous stages. On the other hand, the local multiple alignment uses a local alignment algorithm such as the Smith-Waterman algorithm (Smith, Waterman, & Fitch, 1981) to align only the most conserved motifs. In general, global algorithms give the best results when aligning a family of sequences with an orphan one, aligning equidistant sequences and divergent sequences families. However, they are less precise in the case of large N/C-terminal extensions and internal insertions than local algorithms (Thompson, Plewniak, & Poch, 1999). Many other progressive algorithms have been developed in the last years, like FAMSA (Deorowicz, Debudaj-Grabysz, & Gudyś, 2016) which uses a quick technique for pairwise similarities to create a phylogenetic tree and build alignment faster.

Iterative Alignments

Iterative alignment methods are based on producing an initial alignment and a series of iterations are applied to refine it until no improvement can be applied. Iterative alignment methods are based on the idea that the solution can be made from an already existing suboptimal one.

For example, PRRP (Gotoh, 1994) and MUSCLE (Edgar, 2004) optimize a progressive global alignment by dividing the sequences into two groups using a profile-to-profile alignment algorithm. Dialign (Morgenstern, 2004) and Dialign-TX (Subramanian, Kaufmann, & Morgenstern, 2008) use local information to guide overall alignment. HMMT (Eddy, 1995), a Hidden Markov Model, uses simulated annealing and dynamic programming to create suboptimal alignment. In the last years, several other algorithms have been developed. We find for example the use of genetic algorithm as in (Amiroch, Pradana, Irawan, & Mukhlash, 2019), multiobjective genetic algorithm like in (Chowdhury & Garai, 2017), multiobjective evolutionary algorithms as in (Rubio-Largo, Vanneschi, Castelli, & Vega-Rodríguez, 2018) and using a tabu search algorithm as in (Mehenni, 2015). Most of these algorithms offer the possibility of using them integrally or partially into another alignment process.

Globally, MSA techniques are time-consuming, and with the appearance of large biological datasets, it has become essential to think about how to reduce execution time.

IMPROVING THE SPEED OF MSA

All the techniques proposed to resolve the MSA problem are CPU-time consuming. The focus of recent research has been on how to improve the speed of MSA solvers without losing accuracy. To solve this problem, different approaches were developed that can be categorized into two classes. The first is based on the use of parallelism through hardware approach. Some methods use shared-memory and distributed memory computers, like ClustalW-MPI (K.-B. Li, 2003) and Parallel T-Coffee (Zola, Yang, Rospondek, & Aluru, 2007). Church et al (Church et al., 2011) presented a design of MSA algorithms

on supercomputers with parallel CPUs and distributed memory. Another parallelization of MAFFT applicable to thousands of sequences is presented in (Nakamura, Yamada, Tomii, & Katoh, 2018). On the other hand, graphical processing units are used to accelerate MSA programs, like GPU-Blast (Vouzis & Sahinidis, 2011), G-MSA (Blazewicz, Frohberg, Kierzyńska, & Wojciechowski, 2013), and we can cite the recent work of (Liu, Cui, & Zhao, 2019) based on GPU accelerated sequence alignment and the use of multiple GPUs.

The second class consists of the use of parallelism through software approach by using parallel programming models. In this section, four subclasses may be suggested:

The first one is a parallel approach in the calculation of score matrix. This approach was used in (Zafalon et al., 2013) where an improvement of 15% in execution time was shown.

The second one is a Pipeline approach; Agarwal and Rizvi (Agarwal & Rizvi, 2009) have proposed a technique with two stages pipeline that can improve the complexity of the problem. Then, a novel multi-alignment pipeline for high-throughput sequencing data is presented by Huang et al (Huang, Holt, Kao, McMillan, & Wang, 2014).

The third one is a parallel approach with a dynamic algorithm. In this subclass, the techniques are based on the parallelization of the optimal algorithms known in the field, for example, parallelization of the Needleman & Wunsch algorithm by Naveed (Naveed, Siddiqui, & Ahmed, 2005), the parallelization of the Smith & Waterman algorithm by Dohi et al (Dohi, Benkridt, Ling, Hamada, & Shibata, 2010), and the parallelization of the optimal technique to solve the MSA by Helal et al on GPU architecture (Helal, El-Gindy, Mullin, & Gaeta, 2008).

The fourth one is a data parallel approach. In recent years, research on this kind of optimization has become very popular. This technique, based on the divide and conquer strategy, suggests three main steps: Cluster, Distribute and Align. A non-hierarchical clustering step is proposed in the first step, then a distribution of the clusters on the different processors, and finally an alignment step to build the MSA. Recently, several works have been proposed based on this technique, the first work is presented by Fahad et al (Saeed & Khokhar, 2009) where a k-mer technique is used to do the clustering, but a significant loss in the quality of alignment has been observed. Several clustering techniques were proposed, such as UCLUST (Edgar, 2010), CD-HIT (W. Li & Godzik, 2006), BlastClust (Dondoshansky & Wolf, 2002) and clustering packages like proposed in (Bruneau et al., 2018). These allowed creating different MSA approaches. Xiangyuan et al (Zhu, Li, & Salah, 2013) proposed a data parallel approach based on the two clustering systems UCLUST and CD-HIT in the clustering step and MUSCLE in the alignment step.

Despite the gain of execution time, existing methods on data parallel strategy are still lacking in precision. This inaccuracy is due to the errors produced during the non-hierarchical clustering. However, it is possible to further improve the accuracy by creating clusters with good quality. In this perspective, this work explores a novel clustering method with a greedy local search algorithm.

THE PROPOSED STRATEGY

The proposed strategy to build MSA is based on three main steps: (1) clustering the sequences depending on the number of processor cores using a greedy local search algorithm, (2) aligning each subset and generating a consensus sequence for each one, and (3) aligning all generated consensus sequences using an alignment algorithm to generate the whole alignment. The summary of the different steps of this approach is shown in *Figure 1*.

Sequence Clustering

Algorithms for clustering biological sequences try to group homologous sequences. They are useful for predicting homology and function of biological sequences, reducing redundancy, comparing data from diverse environments and quantifying ecosystem diversity.

Several methods for clustering biological sequences are currently proposed and can be categorized into two main groups: hierarchical and non-hierarchical clustering techniques. Several popular non-hierarchical algorithms have been created in this field such as: UCLUST (Edgar, 2010), CD-HIT (W. Li & Godzik, 2006), BLASTClust (Dondoshansky & Wolf, 2002) and the clustering package for nucleotide sequences using Laplacian Eigenmaps and Gaussian Mixture Model (Bruneau et al., 2018).

It should be noted that to choose a clustering algorithm, we have in general a set of desired features to be satisfied such as scalability and sensitivity. The use of non-hierarchical clustering algorithms can improve significantly the computation time; nevertheless, it causes a significant loss in the quality of alignment, since they are not made for multiple alignments but rather for detecting the redundancy in the databases.

In the proposed strategy, as shown in *Figure 2* and *Figure 3*, a greedy technique based on a local search algorithm is proposed. This strategy starts by an initial state containing several clusters and then improves them iteratively until a good clustering is obtained. In the following, the proposed strategy is presented in more detail.

Sequences Alignment and Consensus Alignment

We have used MUSCLE, Clustal Omega and Mafft algorithms to align sequences in the clusters as well as in the consensus phase. The developed strategy can be considered as a local search to explore promising alignment with any other aligner. *Figure 1* describes the proposed algorithm called GC-MSA. The algorithm is composed of three main parts. The first part consists of the construction of clusters of good quality called Clusters_best. The second part creates from the obtained clusters a set of consensus sequences and aligns them using a multiple alignment algorithm. The third part consists in the propagation of the gaps found in the consensus-alignment to the clusters and then to the original sequences to generate the multiple alignment.

Algorithm 1 represents the Pseudo-code of the algorithm. The first step consists of selecting a sequence randomly without replacement to be the centroid; a two-dimensional array that contains all the distances between the centroid sequence and all the others is created. By decomposing the previously built table, clusters are created. Each generated cluster is aligned independently and a consensus is generated for each cluster. After that, the distance between the sequences of each cluster and their consensus is computed. At the end, the overall distance $\sum_{i=1}^{NbCores} D(i)$ is calculated. If this distance is less than the best distance, the new clustering is better than the existing one, and therefore we keep it. The previous operations are repeated several times (depending on the user's choice) to get out a good clustering used in the next step of the multiple alignment of clusters consensus, and thus an MSA_consensus is generated. At the end, gaps generated during the consensus alignment are propagated in the original clusters of each consensus and afterwards in the original sequences, and thus the multiple alignment of all sequences is obtained.

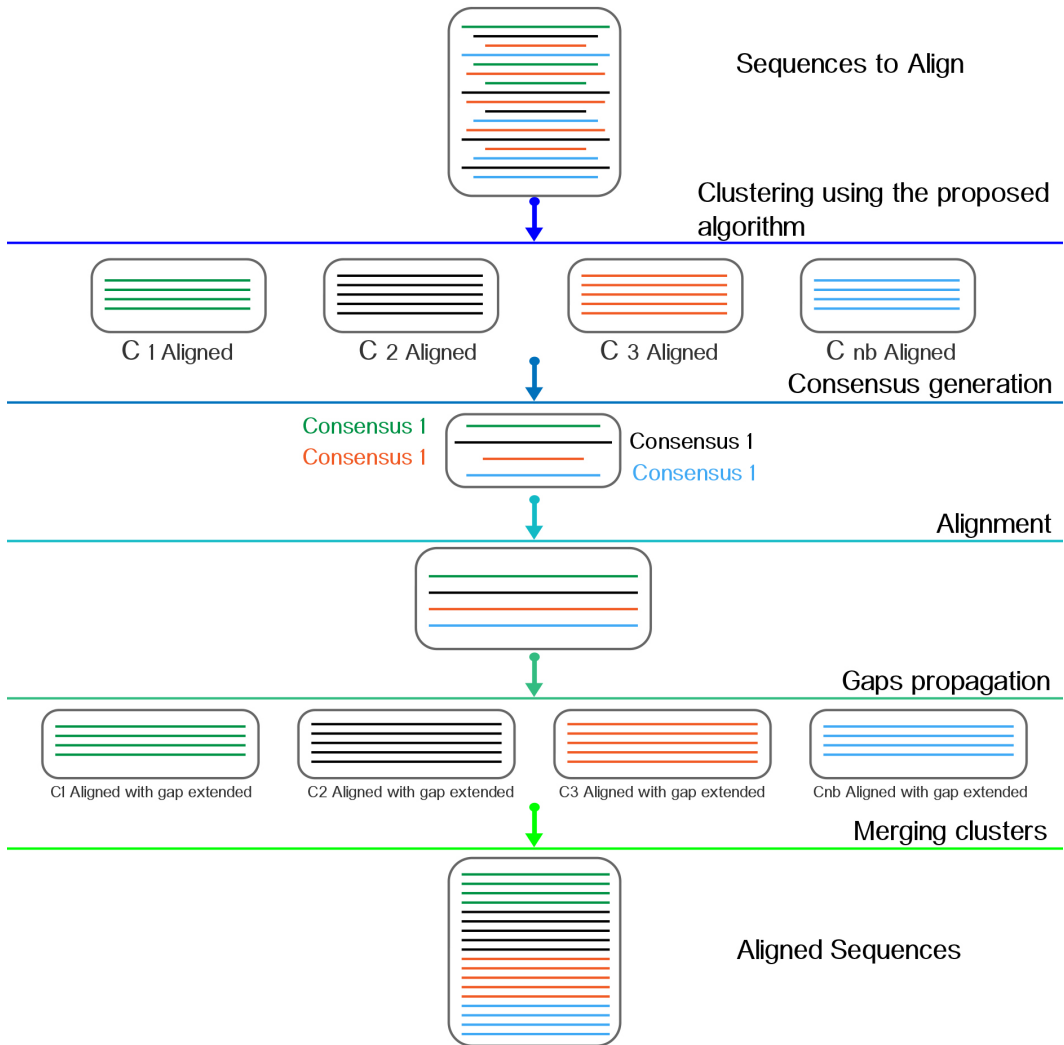
MapReduce GC-MSA

In order to improve the execution time, the distribution of data and calculation was proposed using a version of the algorithm implemented in MapReduce model as shown in *Figure 4*. The mapper represents the clusters creation phase, this one can be repeated several times until a good clustering is achieved as proposed in *Algorithm 1*, each cluster will be deployed and aligned separately on a node. The reducer represents the gap propagation phase to generate the whole global alignment.

EXPERIMENTAL RESULTS AND DISCUSSION

The proposed approach is implemented in MATLAB R2014b and tested on an Intel Xeon ES2620 with 6 cores, running at 2.00 GHz, with caches (L1D-Cache 32 KB, L1I-Cache 32 KB, L2-Cache

Figure. 1. Summary of GC-MSA algorithm



256 KB, L3-Cache 15MB) and with 16 GB DDR3 memory. Several experiments were made to study the gain provided by the proposed strategy.

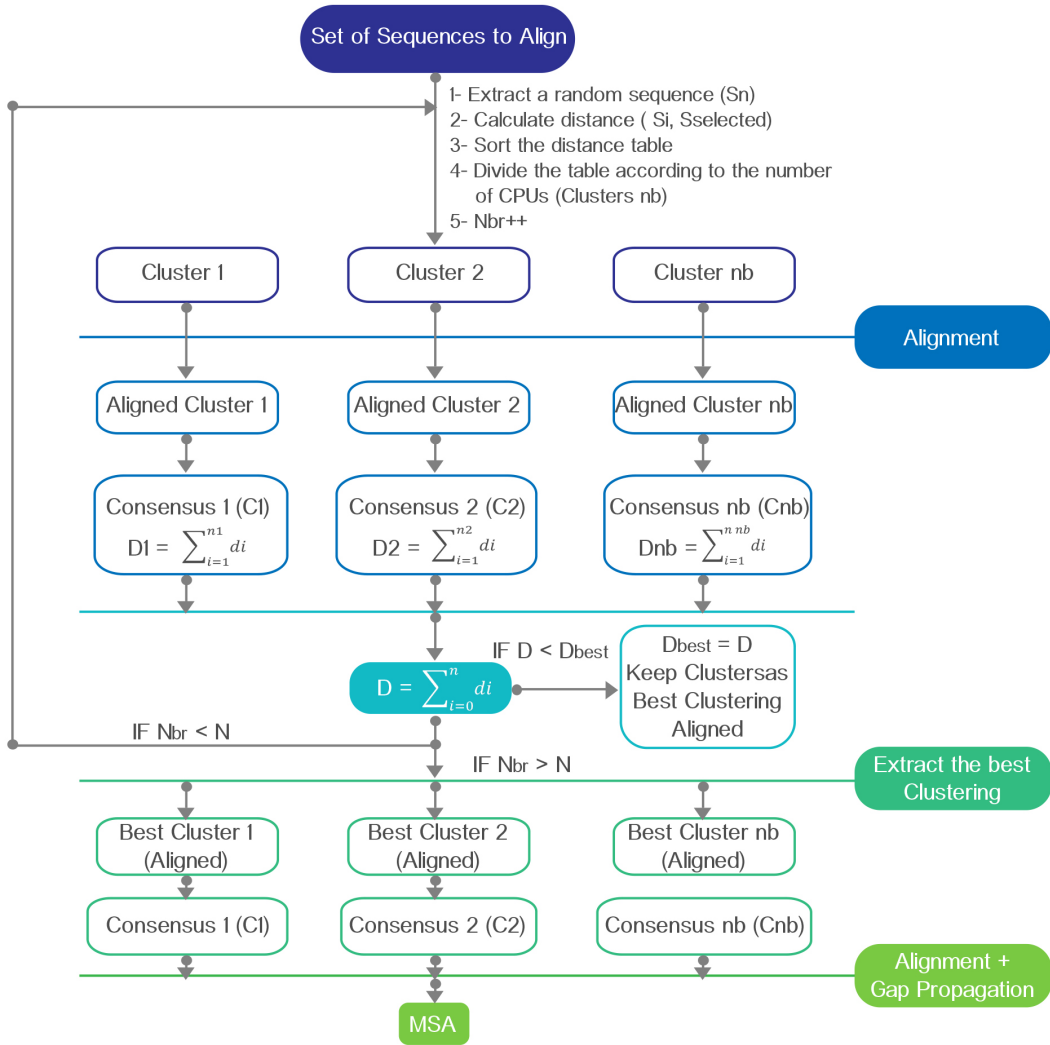
Quality of Alignment

The performance of the proposed approach has been tested on a set of protein sequence alignment benchmarks which are BALIBASE v3, SABRE, PREFAB v4 and OXBENCH. Results are compared with other well-known MSA algorithms such as MAFFT v6.603, MUSCLE and Clustal Omega.

To measure the quality of the alignment, the program qScore (Edgar, 2018) is used, the program outputs the following scores: The PREFAB Q score (also known as the BALiBASE Developer score) and The BALiBASE TC (total column) score.

The BALiBASE benchmark suite is more important than other benchmarks. It contains multiple sequence alignments organized into reference sets representing specific MSA problems, including

Figure. 2. The proposed algorithm for clustering

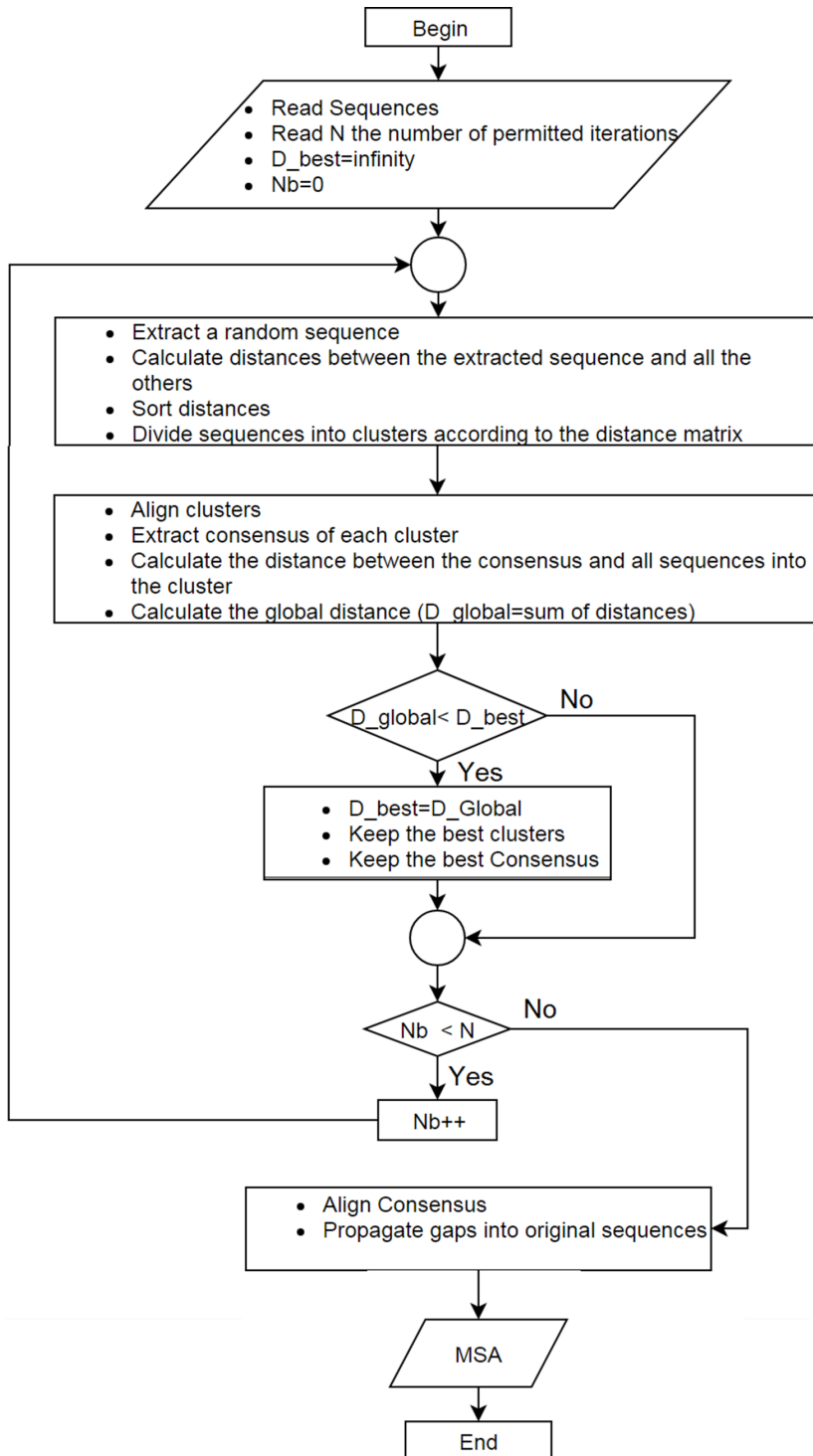


unequal phylogenetic distributions, small number of sequences, unequal phylogenetic distributions, inverted domains and transmembrane regions and large N/C-terminal extensions or internal insertions.

To assess the performance of the three programs: MUSCLE without clustering, MUSCLE with the proposed strategy GC-MSA(MUSCLE) and MUSCLE using UCLUST (Edgar, 2010) algorithm for clustering. Several instances taken from the BAliBASE 3.0 benchmark are used. The results of this experiment are presented in *Table 1* and *Table 2*. It shows clearly the effectiveness of using the proposed strategy to perform MSA.

Moreover, the performance of the proposed strategy integrated in MUSCLE algorithm (GC-MSA-MUSCLE) and Clustal Omega algorithm (GC-MSA-Clustal Omega) is analyzed by comparing the alignments produced with those obtained through other leading alignment techniques. *Table 3* summarizes the overall measurement results. In terms of alignment quality, it is clearly observed that our technique gives good quality results compared to the other programs. In fact, it can be noted that

Figure. 3. The flowchart of the proposed algorithm



Algorithm 1 PSEUDO-CODE OF GC-MSA

Input:
Set of sequences {S1,S2,...,Sn}
Number of iterations permitted
Output:
Multiple sequence alignment
Variables:
Sequences: table of the initial set of sequences;
Consensus_best: table of consensus sequences;
Clusters: structure of clusters;
Clusters_best: structure of the best clusters found;
NbCores:=Number_of_cores_processor;
N: integer, to be defined by users;
begin
Nb:=0;
D_best:=INFINITY;
NbSeq:=Size(Sequences);
while Nb<N
seq:=RandomExtractSeq(Sequences);
for i:=1..NbSeq
T[i]:=calculateDistance(seq, Sequences);
end_for
Sort(T);
Clusters:=creatClustersFromTable(T);
parallel_For i:=1..NbCores
multiAlign(Clusters(i));
consensus[i]:=generateConsensus(Clusters(i));
D[i]:=calculateDistance(consensus[i],Clusters(i));
end_parallel_for
D_global:=SUM(Distance);
if D_global < D_best
D_best:= D_global;
Clusters_best:= Clusters;
end_if
Nb++;
end_while
parallel_For i:=1..NbCores
Consensus_best[i]:=generateConsensus(Clusters_best(i));
end_parallel_for

continued on following page

Algorithm 1. Continued

Input:
MSA_consensus:=multiAlign(Consensus_best);
Gap_propagation(MSA_consensus, Sequences);
end.
Explanation of the functions used in the algorithm
RandomExtractSeq: randomly extract a sequence from a set
calculateDistance: creates a table containing the distances between the chosen sequence and all the others using Needleman-wunsch algorithm.
Sort: sort a table according to the distances.
creatClustersFromTable: creates clusters from the table containing distances.
multiAlign: Align multiple sequences using an Aligner.
generateConsensus: generate a consensus sequence from a MSA.
Gap_propagation: propagates the gaps found inside a sequence into a set of sequences.

Figure 4. The proposed MapReduce model for GC-MSA

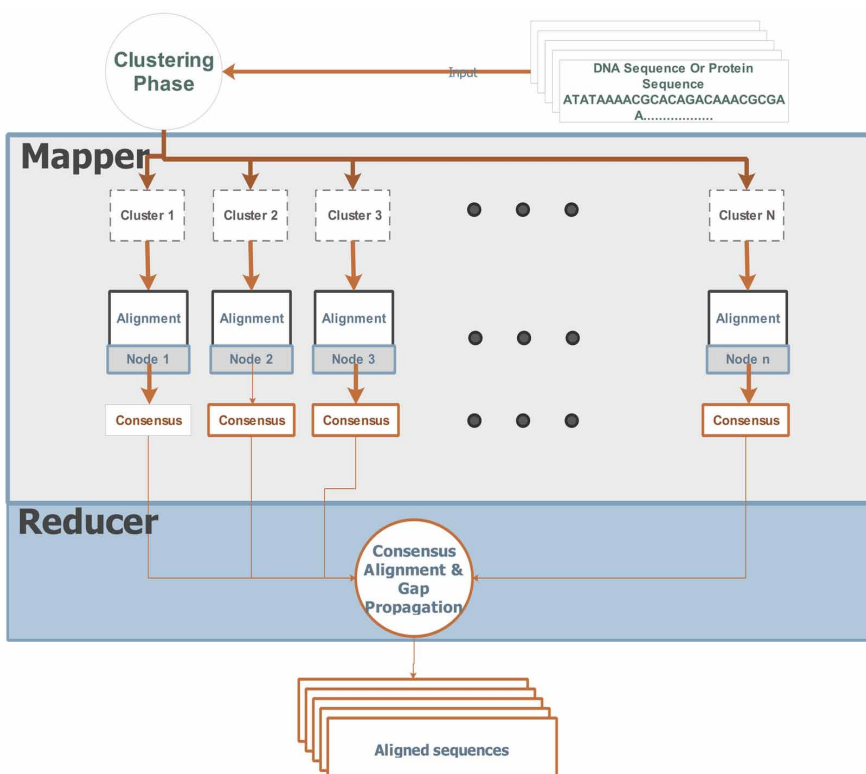


Table 1. Average Q/TC Scores For GC-MSA Implementing Muscle Vs Muscle Algorithm, On BALIBASE 3.0

Algorithms	RV 11	RV 12	RV 20	RV 30	RV 40	RV 50	AVR
GC-MSA (MUSCLE)	0.576 0.500	0.915 0.819	0.891 0.367	0.823 0.581	0.869 0.447	0.835 0.611	0.818 0.554
MUSCLE (UCLUST)	0.438 0.208	0.839 0.629	0.820 0.181	0.652 0.137	0.698 0.255	0.727 0.275	0.696 0.281
MUSCLE	0.497 0.322	0.834 0.696	0.860 0.336	0.686 0.309	0.715 0.373	0.731 0.442	0.721 0.413

Table 2. Average Q/TC Scores For GC-MSA Implementing Muscle Vs Muscle Algorithm, On Different Benchmarks

Algorithms	Sabre	Sabrem	Prefab4	Oxm
GC-MSA (MUSCLE)	0.550 0.355	0.700 0.550	0.650 0.650	1.000 0.899
MUSCLE	0.414 0.215	0.676 0.488	0.610 0.610	0.992 0.893

Table 3. Average Q/TC Scores For Several Alignment Algorithms, On BALIBASE 3.0 Data-Set

Algorithms	RV 11	RV 12	RV 20	RV 30	RV 40	RV 50	AVR
GC-MSA MUSCLE	0.576 0.500	0.915 0.819	0.891 0.367	0.823 0.581	0.869 0.447	0.835 0.611	0.818 0.554
MUSCLE	0.497 0.322	0.834 0.696	0.860 0.336	0.686 0.309	0.715 0.373	0.731 0.442	0.721 0.413
GC-MSA Clustal Omega	0.634 0.421	0.929 0.828	0.942 0.490	0.867 0.579	0.901 0.583	0.862 0.537	0.856 0.573
Clustal Omega	0.590 0.362	0.906 0.794	0.912 0.453	0.863 0.570	0.866 0.579	0.840 0.533	0.830 0.549
GC-MSA Mafft	0.530 0.440	0.826 0.787	0.625 0.450	0.709 0.514	0.742 0.747	0.866 0.771	0.716 0.618
Mafft	0.649 0.411	0.937 0.844	0.927 0.461	0.862 0.588	0.910 0.573	0.899 0.595	0.864 0.579

Table 4. Average Running Time (Seconds) Comparison

Number of sequences	Running time (s)					
	GC-MSA-MUSCLE	MUSCLE	GC-MSA-Clustal Omega	Clustal Omega	GC-MSA-Mafft	Mafft
100	320	161	174	82	272	87
150	460	170	146	103	311	110
200	679	259	191	111	429	136
250	964	328	226	146	501	168
300	1360	458	262	153	618	188
350	1708	789	304	157	824	200
400	2128	1046	337	187	1028	291
450	2146	1064	361	215	1140	464

Table 5. Average Running Time Using Parallelism on A Multicore Processor

Number of sequences	Running time (s)		
	2 Cores	4 Cores	6 Cores
100	287	281	272
150	367	339	311
200	502	454	429
250	597	561	501
300	842	693	618
350	1058	910	824
400	1268	1112	1028
450	1630	1386	1140

Table 6. Average Running Time Using MAPREDUCE Version On Apache Spark

Number of sequences	Running time (s)					
	GC-MSA-MUSCLE	MUSCLE	GC-MSA-Clustal Omega	Clustal Omega	GC-MSA-Mafft	Mafft
100	7	16	4	8	5	8
150	10	18	3	12	7	12
200	15	29	5	16	12	14
250	25	32	7	17	12	18
300	36	48	7	16	17	21
350	44	80	8	17	20	20
400	53	100	9	20	23	32
450	50	107	9	25	30	51

the proposed algorithm gives good alignment results compared to the other algorithms used in the field especially for sequences that can be assembled into families as BALiBASE 3.0 - RV30.

Execution Time

To measure the running time, an experiment was made based on large datasets containing benchmarks sets of sequences generated by GenRGenS (Ponty, Termier, & Denise, 2006) on profiles of real sequences derived from BALiBASE in which the length of a sequence varies from 500 to 2000. Both the sequences number and their lengths have an important consequence on the time of execution of the aligner.

To measure the running time, MUSCLE algorithm is used in our technique. The solution is compared with those of most used aligner programs such as MUSCLE, Clustal Omega and Mafft. *Table 4* shows the difference in the execution time between our technique and those used frequently.

Table 4 shows that the proposed strategy takes more times compared to other programs. In fact, the search for a good clustering needs more time but at the same time, it improves considerably the result

as it is shown in the *Table 3*. One of the causes of this slowness is the use of Matlab language known by its slowness compared to other compiled languages such as C/C++. The speed of the algorithm can be increased considerably by choosing other languages like those used by other programs used in this comparison. Moreover, the runtime could be enhanced by using more specific local search.

Moreover, the performance is analyzed in terms of execution time and scalability in the case of the use of parallelism. Experiments are performed on datasets containing the eight test sets used previously and Mafft as an aligner.

In *Table 5*, we can see that the execution time can be reduced by increasing the cores number. This is because the alignment of clusters can be executed in an independent and a parallel way. Clusters can be aligned concurrently by using all processor cores. Since the performance of multi-core computer grows faster, results would be more encouraging and competitive.

The proposed algorithm has significantly improved the alignment quality. However, this method is time consuming. Since the parallel approach on a multicore CPU had no considerable improvement, the MapReduce approach executed on the Apache Spark platform is proposed and had a better execution time scoring. MapReduce was executed on a grid computing with 32 nodes, with Intel® Xeon® E5-2699A v4 CPU 2.60-3.00 GHz 55 MB Cache memory. Table 6 shows this gain from this release.

CONCLUSION

In this paper, a new strategy to tackle the MSA problem is developed based on the divide and conquer approach. The preliminary step consists to divide all the sequences into subsets and then align each subset with an aligner and at the end, create an MSA for all sequences. This strategy includes a new greedy method for clustering sequences based on local search algorithm. To measure the alignment quality produced by the proposed strategy, several benchmarks known in the field like as BALiBASE 3.0, SABRE, OXM and Prefab are used. The results obtained indicate that the proposed approach is able to give good quality alignments with small increase in running time. To reduce the execution time, the distribution was proposed where the MapReduce model was used and showed a significant improvement. Besides, the proposed approach can provide an extensible platform for improving other alignment programs. Unfortunately, creating clusters without High Performance Computing can be a time consuming processing. To deal with this problem, it would be interesting to introduce a k-mer algorithm to compare sequences and a profile-profile alignment method to align clusters. On the basis of the promising findings presented in this paper, we work on the improvement of the execution time and the application of the proposed architecture for solving other bioinformatics problems.

REFERENCES

- Agarwal, P., & Rizvi, S. (2009). Solving sequence alignment problem using pipeline approach. Bharati Vidyapeeth's Institute of Computer Applications and Management, 107.
- Amiroch, S., Pradana, M. S., Irawan, M., & Mukhlash, I. (2019). A Simple Genetic Algorithm for Optimizing Multiple Sequence Alignment on the Spread of the SARS Epidemic. *The Open Bioinformatics Journal*, 12(1), 30–39. doi:10.2174/1875036201912010030
- Blazewicz, J., Frohberg, W., Kierzyńska, M., & Wojciechowski, P. (2013). G-MSA—A GPU-based, fast and accurate algorithm for multiple sequence alignment. *Journal of Parallel and Distributed Computing*, 73(1), 32–41. doi:10.1016/j.jpdc.2012.04.004
- Bruneau, M., Mottet, T., Moulin, S., Kerbirou, M., Chouly, F., Chretien, S., & Guyeux, C. (2018). A clustering package for nucleotide sequences using Laplacian Eigenmaps and Gaussian Mixture Model. *Computers in Biology and Medicine*, 93, 66–74. doi:10.1016/j.compbiomed.2017.12.003 PMID:29288886
- Chowdhury, B., & Garai, G. (2017). A review on multiple sequence alignment from the perspective of genetic algorithm. *Genomics*, 109(5-6), 419–431. doi:10.1016/j.ygeno.2017.06.007 PMID:28669847
- Church, P. C., Goscinski, A., Holt, K., Inouye, M., Ghoting, A., Makarychev, K., & Reumann, M. (2011). *Design of multiple sequence alignment algorithms on parallel, distributed memory supercomputers*. Paper presented at the Engineering in Medicine and Biology Society, EMBC, 2011 Annual International Conference of the IEEE. doi:10.1109/IEMBS.2011.6090208
- Deorowicz, S., Debudaj-Grabysz, A., & Gudyś, A. (2016). FAMSA: Fast and accurate multiple sequence alignment of huge protein families. *Scientific Reports*, 6(1), 33964. doi:10.1038/srep33964 PMID:27670777
- Dohi, K., Benkridt, K., Ling, C., Hamada, T., & Shibata, Y. (2010). *Highly efficient mapping of the Smith-Waterman algorithm on CUDA-compatible GPUs*. Paper presented at the Application-specific Systems Architectures and Processors (ASAP), 2010 21st IEEE International Conference on. doi:10.1109/ASAP.2010.5540796
- Dondoshansky, I., & Wolf, Y. (2002). *Blastclust (ncbi software development toolkit)*. NCBI.
- Eddy, S. R. (1995). *Multiple alignment using hidden Markov models*. Paper presented at the Ismb.
- Edgar, R. C. (2004). MUSCLE: Multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32(5), 1792–1797. doi:10.1093/nar/gkh340 PMID:15034147
- Edgar, R. C. (2010). Search and clustering orders of magnitude faster than BLAST. *Bioinformatics (Oxford, England)*, 26(19), 2460–2461. doi:10.1093/bioinformatics/btq461 PMID:20709691
- Edgar, R. C. (2018). *A quality scoring program*. <http://www.drive5.com/qscore/>
- Feng, D.-F., & Doolittle, R. F. (1987). Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4), 351–360. doi:10.1007/BF02603120 PMID:3118049
- Gotoh, O. (1994). Further improvement in methods of group-to-group sequence alignment with generalized profile operations. *Bioinformatics (Oxford, England)*, 10(4), 379–387. doi:10.1093/bioinformatics/10.4.379 PMID:7804871
- Helal, M., El-Gindy, H., Mullin, L., & Gaeta, B. (2008). *Parallelizing optimal multiple sequence alignment by dynamic programming*. Paper presented at the Parallel and Distributed Processing with Applications, 2008. ISPA'08. International Symposium on. doi:10.1109/ISPA.2008.93
- Hirosawa, M., Totoki, Y., Hoshida, M., & Ishikawa, M. (1995). Comprehensive study on iterative algorithms of multiple sequence alignment. *Bioinformatics (Oxford, England)*, 11(1), 13–18. doi:10.1093/bioinformatics/11.1.13 PMID:7796270
- Huang, S., Holt, J., Kao, C.-Y., McMillan, L., & Wang, W. (2014). A novel multi-alignment pipeline for high-throughput sequencing data. *Database (Oxford)*, 2014(0), bau057. doi:10.1093/database/bau057 PMID:24948510
- Li, K.-B. (2003). ClustalW-MPI: ClustalW analysis using distributed and parallel computing. *Bioinformatics (Oxford, England)*, 19(12), 1585–1586. doi:10.1093/bioinformatics/btg192 PMID:12912844

- Li, W., & Godzik, A. (2006). Cd-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics (Oxford, England)*, 22(13), 1658–1659. doi:10.1093/bioinformatics/btl158 PMID:16731699
- Liu, Y., Cui, H., & Zhao, R. (2019). Fast Acquisition of Spread Spectrum Signals using Multiple GPUs. *IEEE Transactions on Aerospace and Electronic Systems*, 55(6), 3117–3125. doi:10.1109/TAES.2019.2902695
- Mehenni, T. (2015). *Multiple guide trees in a tabu search algorithm for the multiple sequence alignment problem*. Paper presented at the IFIP International Conference on Computer Science and its Applications. doi:10.1007/978-3-319-19578-0_12
- Morgenstern, B. (2004). DIALIGN: Multiple DNA and protein sequence alignment at BiBiServ. *Nucleic Acids Research*, 32(suppl_2), W33–W36. doi:10.1093/nar/gkh373 PMID:15215344
- Nakamura, T., Yamada, K. D., Tomii, K., & Katoh, K. (2018). Parallelization of MAFFT for large-scale multiple sequence alignments. *Bioinformatics (Oxford, England)*, 34(14), 2490–2492. doi:10.1093/bioinformatics/bty121 PMID:29506019
- Naveed, T., Siddiqui, I. S., & Ahmed, S. (2005). *Parallel needleman-wunsch algorithm for grid*. Paper presented at the DIALIGN-TX ngs of the PAK-US International Symposium on High Capacity Optical Networks and Enabling Technologies (HONET 2005), Islamabad, Pakistan.
- Needleman, S. B., & Wunsch, C. D. (1970). 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. doi:10.1016/0022-2836(70)90057-4 PMID:5420325
- Notredame, C. (2002). Recent progress in multiple sequence alignment: A survey. *Pharmacogenomics*, 3(1), 131–144. doi:10.1517/14622416.3.1.131 PMID:11966409
- Ponty, Y., Termier, M., & Denise, A. (2006). GenRGenS: Software for generating random genomic sequences and structures. *Bioinformatics (Oxford, England)*, 22(12), 1534–1535. doi:10.1093/bioinformatics/btl113 PMID:16574695
- Rubio-Largo, Á., Vanneschi, L., Castelli, M., & Vega-Rodríguez, M. A. (2018). Multiobjective characteristic-based framework for very-large multiple sequence alignment. *Applied Soft Computing*, 69, 719–736. doi:10.1016/j.asoc.2017.06.022
- Saeed, F., & Khokhar, A. (2009). A domain decomposition strategy for alignment of multiple biological sequences on multiprocessor platforms. *Journal of Parallel and Distributed Computing*, 69(7), 666–677. doi:10.1016/j.jpdc.2009.03.006
- Sievers, F., & Higgins, D. G. (2014). Clustal Omega, accurate alignment of very large numbers of sequences. *Multiple Sequence Alignment Methods*, 105–116.
- Smith, T. F., Waterman, M. S., & Fitch, W. M. (1981). Comparative biosequence metrics. *Journal of Molecular Evolution*, 18(1), 38–46. doi:10.1007/BF01733210 PMID:7334527
- Subramanian, A. R., Kaufmann, M., & Morgenstern, B. (2008). DIALIGN-TX: Greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology; AMB*, 3(1), 6. doi:10.1186/1748-7188-3-6 PMID:18505568
- Thompson, J. D., Higgins, D. G., & Gibson, T. J. (1994). 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. doi:10.1093/nar/22.22.4673 PMID:7984417
- Thompson, J. D., Plewniak, F., & Poch, O. (1999). A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research*, 27(13), 2682–2690. doi:10.1093/nar/27.13.2682 PMID:10373585
- Vouzis, P. D., & Sahinidis, N. V. (2011). GPU-BLAST: Using graphics processors to accelerate protein sequence alignment. *Bioinformatics (Oxford, England)*, 27(2), 182–188. doi:10.1093/bioinformatics/btq644 PMID:21088027
- Wang, L., & Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4), 337–348. doi:10.1089/cmb.1994.1.337 PMID:8790475

Zafalon, G. F., Marucci, E. A., Momente, J. C., Amazonas, J. R., Sato, L. M., & Machado, J. M. (2013). *Improvements in the score matrix calculation method using parallel score estimating algorithm*. Academic Press.

Zhu, X., Li, K., & Salah, A. (2013). A data parallel strategy for aligning multiple biological sequences on multi-core computers. *Computers in Biology and Medicine*, 43(4), 350–361. doi:10.1016/j.combiomed.2012.12.009 PMID:23414778

Zola, J., Yang, X., Rospondek, A., & Aluru, S. (2007). PARALLEL-TCOFFEE: A parallel multiple sequence aligner. *ISCA PDCS*, 7, 248–253.

Rabah Lebsir is a PhD Student at Constantine2 University in Algeria. He is a member of MISC Laboratory. He received his engineering degree in Computer Science from University of Constantine, Algeria. Lebsir works on optimization methods to solve Bioinformatics problems.

Abdesslem Layeb is a Professor in the Department of Computer Science at the University of Constantine, Algeria. He is a member of MISC Laboratory. He received his PhD in Computer Science from the University of Constantine, Algeria. Pr. Layeb is interested in the combinatorial optimization methods and their applications to solve several problems from different domains like transportation problems, Bioinformatics, academic problems, etc.

Fariza Tahi is an Associate Professor at university of Evry. She is a member of AROBAS Team, IBISC Laboratory, her main research interest is bioinformatics and computational biology, notably in silico genome analysis, non-coding RNA secondary structure prediction, prediction at large scale of non-coding RNAs (micro-RNAs, piRNAs, ...), data integration, modeling and simulation of biological processes, and regulatory networks analysis.