A Parallel Multiobjective Metaheuristic for Multiple Sequence Alignment

Álvaro Rubio-Largo[*], Leonardo Vanneschi[*], Mauro Castelli[*], Miguel A. Vega-Rodríguez[†]

December 1, 2017

Abstract

The alignment among three or more nucleotides/amino-acids sequences at the same time is known as Multiple Sequence Alignment (MSA), an NP-hard optimization problem. The time complexity of finding an optimal alignment raises exponentially when the number of sequences to align increases. In this work, we deal with a multiobjective version of the MSA problem where the goal is to simultaneously optimize the accuracy and conservation of the alignment. A parallel version of the Hybrid Multiobjective Memetic Metaheuristics for Multiple Sequence Alignment is proposed. In order to evaluate the parallel performance of our proposal, we have selected a pull of datasets with different number of sequences (up to 1000 sequences) and study its parallel performance against other well-known parallel metaheuristics published in the literature, such as MSAProbs, T-Coffee, Clustal Ω, and MAFFT. The comparative study reveals that our parallel aligner is around 25 times faster than the sequential version with 32 cores, obtaining a parallel efficiency around 80%.

1 Introduction

Multiple Sequence Alignment (MSA) is the process of aligning three or more nucleotides/amino-acids sequences at the same time [1]. The main aim of MSA is to discover common ancestors among biological sequences. The discovery of biological relationship among several sequences is vital for inferring phylogenetic relationships among groups of organisms [4] [7]. Another important goal of MSA is the determination of biological significance among the given sequences [14], therefore, we prioritize the conservation within regions throughout the alignment process. Finally, a proper alignment helps us to detect which regions of a gene are susceptible to mutation and which can have one residue replaced by another without changing the function.

The MSA problem is an NP-complete optimization problem; therefore, different heuristic approaches have been developed for solving this problem in an

[*]NOVA IMS, Universidade NOVA de Lisboa, Portugal. (email: arl@unex.es, lvanneschi@novaims.unl.pt, mcastelli@novaims.unl.pt)

[†]Dept. TC2, University of Extremadura, Spain. (email: mavega@unex.es)

efficient amount of time. In the literature, we find three main groups: *progressive* methods, *consistency-based* methods, and *iterative refinement* methods.

The most representative *progressive* methods are: Clustal W [20], Clustal Ω [19], PRANK [13], and Kalign [11]. These methods start calculating a distance matrix from every pair of the given sequences; then, a guide tree is built by using any hierarchical clustering algorithm, such as Unweighted Pair Group Method with Arithmetic Mean (UPGMA); finally, the alignment is obtained by following the guide tree. The main disadvantage of *progressive* methods relies on the chance of including an inaccurate gap at the beginning that will be propagated to final alignment.

The second group includes those methods based on *consistency*. These approaches construct a database of local and global alignments between each pair of sequences that helps to build an accurate multiple alignment among all the given sequences. Among the most important consistency-based tools are: Tree-based Consistency Objective Function For alignment Evaluation (T-Coffee) [15], PROBabilistic CONSistency-based multiple sequence alignment (ProbCons) [3], and MSAProbs [12].

Finally, we find the *iterative refinement* tools. The methodology followed by these tools starts by performing a progressive alignment and then, they iterate with the aim of correcting any possible inaccurate gap inserted in the progressive construction stage. The refinement process is repeated until no further improvements are found or until a predefined number of iterations is reached. Among the most widely-used *iterative refinement* methods, we find: MUltiple Sequence Comparison by Log-Expectation (MUSCLE) [5] and Multiple Alignment using Fast Fourier Transform (MAFFT) [10].

In this paper, we propose the use of Multiobjective Optimization and Evolutionary Computation in order to optimize simultaneously quality and consistency of the final alignment. We have already applied these techniques to other optimization problems in the Bioinformatics [8] and Telecommunication fields [17], [16]. A memetic metaheuristic has been chosen for this purpose, the Shuffled Frog-Leaping optimization Algorithm (SFLA) [6], which is based on the evolution of memes carried by the interactive individuals, and a global exchange of information among themselves. The traditional SFLA has been modified for optimizing multiple objective functions simultaneously and hybridized with a local search procedure. We refer to it as a Hybrid Multiobjective Memetic Metaheuristic for the Multiple Sequence Alignment (H4MSA).

Given a set of k non-aligned sequences where the length of the largest sequence is L, the time and space complexity for solving the MSA problem is $O(k2^kL^k)$ [22]; therefore, to find an optimal alignment of a large number of sequences ($\sim$1000 sequences) becomes computationally intractable. Some of the aforementioned tools allow parallelism: Clustal Ω, T-Coffee, MSAProbs, and MAFFT; however, it is still a challenge to develop accurate methods that provide higher parallel efficiencies for aligning very large sets of sequences. The main contribution of this paper is an efficient parallel version of H4MSA for aligning very large sets of sequences accurately.

The rest of the paper is organized as follows. Section 2 formulates the Multiple Sequence Alignment problem. A detailed description of the parallel H4MSA algorithm is presented in Section 3. Section 4 is devoted to compare the alignment accuracy and parallel efficiency of H4MSA with other parallel approaches published in the literature. Finally, the conclusions and future works are presented in Section 5.

2 Multiple Sequence Alignment Problem

Given a set of sequences S: $\{s_1, s_2, \ldots s_k\}$ of lengths $|s_1|$, $|s_2|$, …, $|s_k|$ defined over an alphabet Σ, such as $\Sigma_{nucleotides}$=$\{A, C, G, T\}$ or $\Sigma_{aminoacids}$=$\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$.

A multiple sequence alignment of S is defined as S': $\{s'_1, s'_2, \ldots, s'_k\}$, where the length of the k sequences is exactly the same. Note that, S' is defined over the same alphabet as S (Σ) with an additional gap symbol ($-$); so, S' is defined over the alphabet $\Sigma \cup \{-\}$.

In this way, a multiple alignment is obtained by adding gaps to the sequences of S so that their lengths become the same. It can be seen as a matrix representation where the rows are sequences and the columns represent aligned symbols. Each column of an alignment must contain at least one symbol of Σ, in other words, a column with all gaps is not allowed. An example of multiple sequence alignment may be:

- Unaligned Sequences (input):

  s_1: `IHNFPICPI (9)`

  s_2: `KPRFVNSDIHNSPGIFPICPI (21)`

  s_3: `KPRFVNSDIHNVNRYFPGICPI (22)`

  s_4: `KPRFVNSDIHNFPGICPI (18)`

- Aligned Sequences (output):

  s'_1: `--------IHN----FP-I---CPI (25)`

  s'_2: `KPRFVNSDIHN----SPGIFPICPI (25)`

  s'_3: `KPRFVNSDIHNVNRYFPGI---CPI (25)`

  s'_4: `KPRFVNSDIHN----FPGI---CPI (25)`

To find an accurate alignment, we propose the use of multiobjective optimization. Therefore, we search the best solution (alignment) that simultaneously maximizes the weighted sum-of-pairs function with affine gap penalties (WSP, f_1) [9] and the number of Totally Conserved (f_2) columns score [5], [21].

On the one hand, the weighted sum of pairs with affine gaps (WSP, f_1) needs to maximize the following equation:

$$WSP(S') = \sum_{l=1}^{AL} SP(l) - \sum_{i=1}^{k} AGP(s'_i) \tag{1}$$

In equation 1, AL is the alignment length, $SP(l)$ is the sum-of-pairs score of the l^{th} column, which is defined as:

$$SP(l) = \sum_{i=1}^{k-1} \sum_{j=i}^{k} W_{i,j} \times \delta(s'_{i,l}, s'_{j,l}) \tag{2}$$

Note that in equation 2, δ is the substitution matrix used, either Pointed Accepted Mutation (PAM) or Block Substitution Matrix (BLOSUM); and $W_{i,j}$ refers to the sequence weight between sequence s_i and s_j. To compute the weight between two sequences we use the following equation:

$$W_{i,j} = 1 - \frac{LD(s_i, s_j)}{max(|s_i|, |s_j|)} \tag{3}$$

The Levenshtein Distance (LD) between two non-aligned sequences is the minimum number of *insertions*, *deletions* or *substitutions* required to change one sequence into the other.

In equation 1, $AGP(s'_i)$ is the affine gap penalty score of sequence s'_i:

$$AGP(s'_i) = (g_o \times \#gaps) + (g_e \times \#spaces) \tag{4}$$

where g_o is the weight to *open the gap* and g_e is the weight to *extend the gap* with one more space. In this work, we have used the BLOSUM62 substitution matrix, g_o=6, and g_e=0.85.

On the other hand, the number of Totally Conserved (f_2) columns score refers to the number of columns that are completely aligned with exactly the same compound. This objective function needs to be maximized to ensure more conserved regions within the alignment.

The maximum number of columns (alignment length) was limited to:

$$maxLength = \left\lceil \frac{3}{2} * max(|s_1|, |s_2|, \ldots, |s_k|) \right\rceil \tag{5}$$

The choice of 1.5 as a scaling factor allowed the alignment to be 50% longer than the longest sequence in the set. This choice was based on the observation that solutions to common alignment problems rarely contained more than 50% gaps.

3 Parallel H4MSA

The Shuffled Frog-Leaping optimization Algorithm (SFLA) is a memetic meta-heuristic developed by Eusuff and Lansey [6]. The search procedure begins with a random population of frogs (solutions) in a swamp.

The population of frogs is divided into isolated communities (memeplexes) that will evolve independently, allowing different directions within the search space. At each community, the frogs evolve by sharing their ideas with their neighbour frogs. In this way, those frogs with better ideas will share more ideas than those frogs with poor ideas.

In addition, the best frogs of each community will share their ideas with other frogs in different communities. After a number of iterations, the communities of frogs are forced to mix and new communities are formed through a shuffling process in order to accelerate the convergence of the algorithm.

The chromosome representation of a solution in H4MSA is different from the traditional binary representation ('1' indicates gap symbol and '0' indicates a residue). For example, the binary representation of the alignment shown in Section 2 will be:

s'_1: `1111111100011110010111000`

s'_2: `0000000000011110000000000`

s'_3: `0000000000000000000111000`

s'_4: `0000000000011110000111000`

In H4MSA, a solution only stores the *number of groups of gaps* followed by the information of each group: *position of the first gap* and *number of successive gaps* (negative value). The chromosome representation of the example alignment will be:

s'_1: 4, (1,-7), (12,-3), (18), (20,-2)

s'_2: 1, (12,-3)

s'_3: 1, (20,-2)

s'_4: 2, (12,-3), (20,-2)

Given a set of k unaligned sequences (S), m memeplexes (number of communities) with n frogs per memeplex, a fixed number of evolutionary steps (N), and a stopping criterion, the procedure of H4MSA is:

1. Generate and evaluate $m \times n$ random alignments/frogs.
2. Sort the $m \times n$ frogs by alignment quality (f_1) and conservation (f_2). In this work, we have used the *Fast Non-Dominated Sorting* procedure [2].
3. Divide the $m \times n$ frogs into m memeplexes, such the first best frog goes to the first memeplex (Y_1), the second best frog to the second memeplex (Y_2), the m^{th} best frog to the m^{th} memeplex (Y_m), the $m+1^{th}$ best frog to Y_1, and so on.
4. For each memeplex (Y_i)
   (a) Select the local worst (X_{lw}) and local best frog (X_{lb}) of the memeplex.
   (b) X_{lw} learns from X_{lb}, that is to say, X_{lw} replaces a portion of its alignment with information obtained from X_{lb}, generating a new frog (X_{new}). For example, we select the following portion of the local best alignment:

X_{lb}:

s'_1: `---------IHN----FP-I---CPI (26)`

s'_2: `KPRFVN-SDIHN----SPGIFPICPI (26)`

s'_3: `KPRFV-NSDIHNVNRYFPGI---CPI (26)`

s'_4: `KPRFV-NSDIHN----FPGI---CPI (26)`

X_{lw}:

s'_1: `----IHNFPI-------CPI-------- (28)`

s'_2: `K--PRFVNSDI-HNS---PGI-FPICPI (28)`

s'_3: `KP-RFVNSD--IHNVNRYFPGI---CPI (28)`

s'_4: `KP-RFVNSD--IHN---FPGI-CPI--- (28)`

and the resultant new frog (X_{new}) is constructed by taking into account both portions, filling with gaps until the length of all the sequences is exactly the same:

X_{new}:

s'_4: `----IHNFPI------- --------CPI (28)`

s'_1: `K--PRFVNSDI-HNS---PGI-FPI CPI (28)`

s'_3: `KP-RFVNSD--IHNVNRYFPGI--- CPI (28)`

s'_2: `KP-RFVNSD--IHN---FPGI- ---CPI (28)`

(c) Apply the following mutation process to X_{new}:

i. *Move a block*: randomly select a block of gaps/compounds and move it one position towards the left or right.

$\dashrightarrow$ (right)

s'_1: `----IHNFPI---------------CPI (28)`

s'_2: `K--PRFVNSDI-HNS---PGI-FPICPI (28)`

s'_3: `KP-RFVNSD--IHNVNRYFPGI---CPI (28)`

s'_4: `KP-RFVNSD--IHN---FPGI----CPI (28)`

▼

s'_1: `I----HNFPI---------------CPI (28)`

s'_2: `KP--RFVNSDI-HNS---PGI-FPICPI (28)`

s'_3: `KPR-FVNSD--IHNVNRYFPGI---CPI (28)`

s'_4: `KPR-FVNSD--IHN---FPGI----CPI (28)`

ii. *Merge two groups*: randomly select one of the sequences, choose a group of gaps/compounds, and merge it with the closest group.

s'_1: `I- -- -HNFPI- - -------------CPI (28)`

s'_2: `KP--RFVNSDI-HNS---PGI-FPICPI (28)` $\dashleftarrow$

s'_3: `KP R- FVNSD-- I HNVNRYFPGI---CPI (28)`

s'_4: `KP R- FVNSD-- I HN---FPGI----CPI (28)`

▼

s'_1: `I----HNFP I-- -------------CPI (28)`

s'_2: `KPRFVNSDI ---HNS---PGI-FPICPI (28)` $\dashleftarrow$

s'_3: `KPR-FVNSD --I HNVNRYFPGI---CPI (28)`

s'_4: `KPR-FVNSD --I HN---FPGI----CPI (28)`

iii. *Divide a group*: randomly select one of the sequences, choose a group of gaps/compounds, and divided it into two new groups of approximately the same size.

s'_1: `I----HNFP I- --------------CPI (28)`

s'_2: `KPRFVNSDI -- -HNS---PGI-FPICPI (28)`

s'_3: `KPR-FVNSD--IHNVNRYFPGI---CPI (28)` $\dashleftarrow$

s'_4: `KPR-FVNSD -- IHN---FPGI----CPI (28)`

▼

s'_1: `I----HNF P I - --------------CPI (28)`

s'_2: `KPRFVNSD I - - -HNS---PGI-FPICPI (28)`

s'_3: `KPR-FVNS - D - IHNVNRYFPGI---CPI (28)←--`

s'_4: `KPR-FVNS D - - IHN---FPGI----CPI (28)`

iv. *Compact Alignment*: delete those columns with all gaps.

s'_4: `I----HNFPI---------------CPI (28)`

s'_1: `KPRFVNSDI---HNS---PGI-FPICPI (28)`

s'_3: `KPR-FVNS-D-IHNVNRYFPGI---CPI (28)`

s'_2: `KPR-FVNSD--IHN---FPGI----CPI (28)`

▼

s'_4: `I----HNFPI--------------CPI (27)`

s'_1: `KPRFVNSDI--HNS---PGI-FPICPI (27)`

s'_3: `KPR-FVNS-DIHNVNRYFPGI---CPI (27)`

s'_2: `KPR-FVNSD-IHN---FPGI----CPI (27)`

(d) Evaluate X_{new}, if X_{new} is better than X_{lw}, then go to Step 4(j).

(e) Select the global best frog (X_{gb}).

(f) X_{lw} learns from X_{gb}, generating a new frog (X_{new}).

(g) Apply the mutation process to X_{new}.

(h) Evaluate X_{new}, if X_{new} is better than X_{lw}, then go to Step 4(j).

(i) Apply a *Local Search* to X_{lw} and evaluate the new alignment produced. In our local search procedure, we use the fast and accurate Kalign2 [11] with the aim of re-aligning a portion of the input alignment. In the following, we present a step-by-step procedure of the local search:

i. Compute a random position of the alignment and a random size in the range [5-25%] of the alignment length:

s'_1: `I----HNFPI--------------CPI (27)`

s'_2: `KPRFVNSDI--HNS---PGI-FPICPI (27)`

s'_3: `KPR-FVNS-DIHNVNRYFPGI---CPI (27)`

s'_4: `KPR-FVNSD-IHN---FPGI----CPI (27)`

ii. Remove all gaps in the selected portion:

Portion s'_1: `NFPI--`		`NFPI`
Portion s'_2: `SDI--H`	▶	`SDIH`
Portion s'_3: `NS-DIH`		`NSDIH`
Portion s'_4: `NSD-IH`		`NSDIH`

iii. Re-align the portion with Kalign2 [11] method.

Input:				Output:
`NFPI`				`NFPI-`
`SDIH`	▶	Kalign2	▶	`-SDIH`

```
NSDIH                NSDIH
NSDIH                NSDIH
```

iv. The new portion re-aligned by Kalign2 replaces the old portion:

s'_1: `I----H NFPI- ------------CPI (26)`

s'_2: `KPRFVN -SDIH NS---PGI-FPICPI (26)`

s'_3: `KPR-FV NSDIH NVNRYFPGI---CPI (26)`

s'_4: `KPR-FV NSDIH N---FPGI----CPI (26)`

(j) Replace X_{lw} by X_{new} and update the set of non-dominated solutions with X_{new}.

(k) If the maximum number of evolutionary steps (N) has not been reached, then go to Step 4(a). Otherwise, continue with the next memeplex.

5. Merge the frogs from the m memeplexes.

6. If the stopping criterion is satisfied, output the set of non-dominated solutions; otherwise, go to Step 2.

As we can see, the output of H4MSA is a set of non-dominated solutions, that is, a set of solutions that represents a trade-off between alignment quality (f_1) and consistency (f_2). A detailed description of the H4MSA algorithms appears in [18].

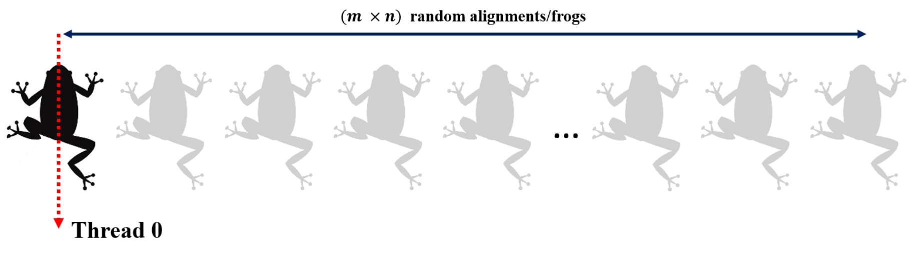
In this work, we propose a parallel scheme of H4MSA for a shared-memory architecture. After studying the computational requirements of H4MSA, we can see that a large amount of time is spent in the initial generation of the population and in the evolving process of each memeplex; therefore, we focus on parallelizing these tasks.

On the one hand, in the generation of the initial population (Step 1), the $m \times n$ random alignments are divided among the available threads. So, if the number of threads is equal to M, each thread will be in charge of generating and evaluating $\frac{m \times n}{M}$ random alignments (frogs). In Figure 1, we can see an illustrative comparison between the sequential and parallel procedure.

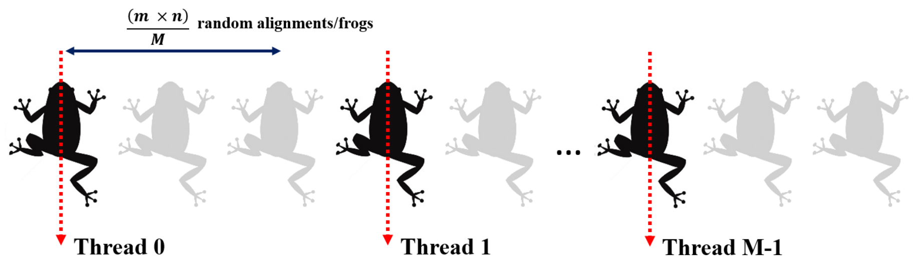
The second and third steps of H4MSA are carried out only by one thread, because the sorting and division process consumes insignificant runtime in comparison with other tasks.

In the fourth step, H4MSA performs m independent evolutionary processes of N iterations. As we can see, the tasks carried out within each iteration of the evolutionary process are computationally expensive: learning process, mutation process, and local search procedure. This loop has been parallelized, but we have defined a critical section when the X_{lw} is replaced by the X_{new} and the set of non-dominated solutions is updated (Step 4(j)).

In the evolving process loop, the workload of each iteration may be different for each thread; therefore, the distribution of iterations is not static, that is, each thread is in charge of performing the consecutive $\frac{N}{M}$ iterations of the evolutionary process, assuming M available threads. On the contrary, in this approach, we use a dynamic distribution scheduling of the N iterations among the threads during runtime.



(a) Sequential



(b) Parallel

Figure 1: Sequential vs. Parallel generation of the initial population in H4MSA.

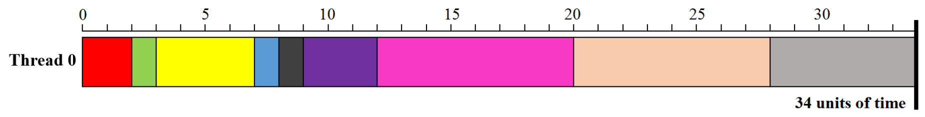
Since H4MSA has been implemented in OpenMP, we have parallelized the evolving of each memeplex by using the `dynamic schedule`. This schedule uses an internal work queue with the loop iterations to each thread. When a thread is finished, it retrieves the next loop iteration from the top of the work queue. The number of iterations performed by each thread is decided during the execution of the algorithm; so, threads may do different number of iterations. In Figure 2, we show the advantages of using a dynamic distribution of the iterations among threads when the workload of iterations is different. As we can see, 51.51% of efficiency is achieved with an static schedule; however, the efficiency increases up to 94.45% when the distribution of the iterations is dynamic.

4 Experimental Results

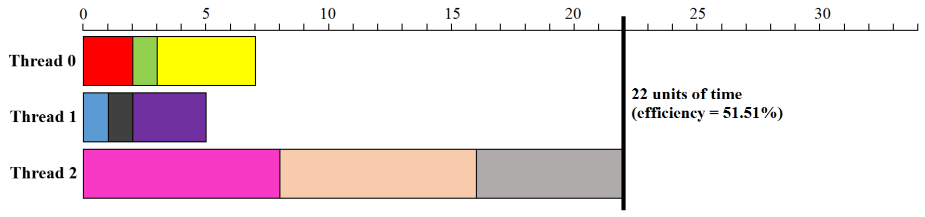
In this section we present a comparative study between our parallel approach (H4MSA) and other multi-threaded MSA approaches published in the literature.

We have compared the multithreaded version of H4MSA with those multiple sequence alignment approaches that allow being run in multi-core environments:

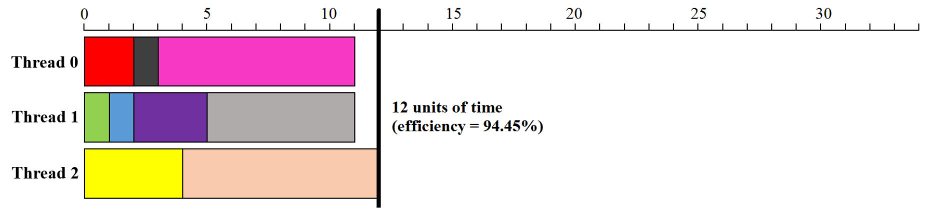
- **MSAProbs** (version 0.9.7) [12]. It is a parallel and accurate approach for MSA. It allows the use of the `-num_threads` flag to specify the number of threads.
- **T-Coffee** (version 11.00.8) [15]. It is a widely used MSA approach in the field. The steps of T-Coffee are multi-threaded by using the `-multi_core` flag, specifying the number of cores to use by the `-n_core` flag.
- **Clustal Ω** (version 1.2.1) [19]. It is the latest addition to the Clustal family. It offers a significant increase in scalability over previous versions,



(a) Sequential



(b) Parallel (static scheduling)



(c) Parallel (dynamic scheduling)

Figure 2: Static vs. dynamic distribution of iterations.

allowing a large number of sequences to be aligned. It also make use of multiple processors by using the `--threads` flag.

- **MAFFT** (version 7.215) [10]. It is a method for rapid multiple sequence alignment based on fast Fourier transform. From version 6.8, MAFFT switches to the multi-core version by simply specifying the number of threads with the `--thread` flag.

The aforementioned approaches were run by using the default parameter configuration.

In H4MSA we found three main parameters: number of memeplexes (m), number of frogs at each memeplex (n), and the number of evolutionary steps (N). In this comparative study, we have used: m=4 (4 memeplexes), n=32 (32 frogs per memeplex), and N=10 (10 evolutionary steps). The stopping criterion is based on the number of fitness evaluations: 50000 evaluations.

The datasets used in this experiments were taken from the HOMFAM benchmark suite [19]. As we can see in Table 1, we have chosen datasets with different number of sequences, from 88 sequences to 1056 sequences, in order to evaluate the performance of the approaches when the number of sequences increases.

In order to extract useful conclusions with a certain level of statistical confidence, 30 independent runs were performed for each approach involved, and the average runtime was used. The architecture selected for conducting these experiments was a 2-processor AMD Opteron$^{\mathrm{TM}}$ Processor 6376 of 16 cores at 2.3GHz and 12MB Cache running Scientific Linux 6.1.

Table 1: Selected HOMFAM datasets

Dataset	#Seqs.	Max. Length	Min. Length
seatoxin	88	50	34
hip	162	73	54
cyt3	379	127	52
rnasemam	492	140	62
TNF	551	154	33
profilin	682	161	32
ricin	740	241	44
trfl	830	367	18
ltn	1056	267	24

Table 2: Average conservation score in 30 independent runs

Dataset	H4MSA	MSAProbs	T-Coffee	Clustal Ω	MAFFT
seatoxin	**704**	688	686	681	689
hip	**605**	590	595	592	594
cyt3	**485**	347	357	367	389
rnasemam	**572**	547	550	546	550
TNF	**527**	462	466	463	467
profilin	**604**	543	554	541	551
ricin	**576**	472	493	477	485
trfl	**588**	537	542	544	546
ltn	**584**	485	501	499	501

For measuring the performance of the parallel approaches, we have used two well-known metrics: *speedup* and *efficiency*. Amdahl's Law states that the performance improvement to be gained from using some faster mode of execution is limited by the fraction of the time the faster mode can be used. Therefore, Amdahl's Law defines the *speedup* that can be gained by using a particular feature. In a more formal way, let T_M be the runtime for an algorithm using M threads and T_1 the runtime of the sequential version, the speedup reports us how much faster an algorithm will run as opposed to the sequential version. The *efficiency* is computed dividing the obtained speedup with M threads by the number of threads used (M).

In order to determine the accuracy of each MSA method, we have evaluated the level of conservation with the BLOSUM62 substitution matrix. Therefore, the alignment obtained by each approach for each data set was scored by using the `+evaluate blosum62mt` action provided by T-Coffee. Note that, a higher score implies better alignment accuracy.

On the one hand, in Table 2, we present the conservation score obtained by H4MSA, MSAProbs, T-Coffee, Clustal Ω, and MAFFT. In Figure 3, we present a visual comparison among the five approaches in terms of conservation score. As we can see in Figure 3, the alignment accuracies obtained by H4MSA in all the datasets tested are better than the well-known approaches. In addition, if we focus on the largest dataset (*ltn*, 1056 sequences), we observe an average conservation improvement around 14.98%. Therefore, we can conclude that

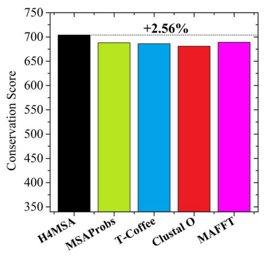
Table 3: Runtime spent by the sequential version of each aligner (in seconds)

	#Seqs.	H4MSA	MSAProbs	T-Coffee	Clustal Ω	MAFFT
seatoxin	88	11	5.3	16	0.2	0.7
hip	162	45	55	182	0.7	4.1
cyt3	379	445	1385	4470	6.3	221
rnasemam	492	566	2095	5460	4.5	104
TNF	551	741	4654	10111	7.5	144
profilin	682	1165	6574	22007	9.4	432
ricin	740	2293	19522	59963	20	766
trfl	830	4146	24713	50496	32	1297
ltn	1056	5787	45554	140579	41	3982

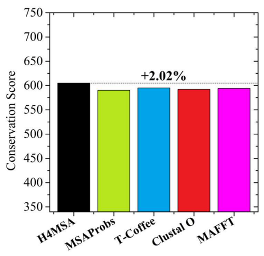
Table 4: Speedup and Efficiency (%) obtained by the multi-threaded MSA approaches.

	H4MSA		MSAProbs		T-Coffee		Clustal Ω		MAFFT	
#Cores	*S*	*E*	*S*	*E*	*S*	*E*	*S*	*E*	*S*	*E*
2	1.96	98.21	1.91	95.46	1.65	82.26	1.10	54.79	1.61	80.44
4	3.71	92.82	3.55	88.66	2.42	60.52	1.18	29.49	2.53	63.32
8	6.87	85.88	6.43	80.36	3.09	38.59	1.23	15.39	3.58	44.76
16	12.82	80.13	10.92	68.28	3.67	22.96	1.29	8.04	4.03	25.18
32	24.98	78.12	17.38	54.33	4.58	14.30	1.32	4.14	4.30	13.42

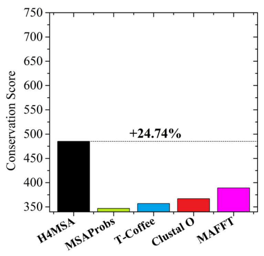
S: Average speedup in the nine datasets of HOMFAM.
E: Average efficiency (%) in the nine datasets of HOMFAM.



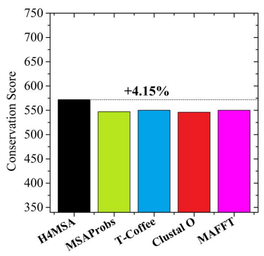
(a) seatoxin (88)



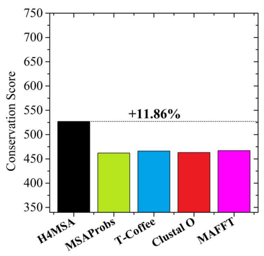
(b) hip (162)



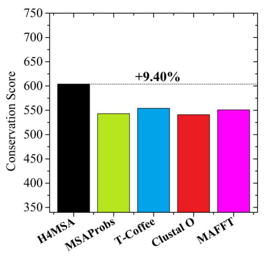
(c) cyt3 (379)



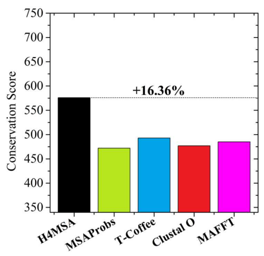
(d) rnasemam (492)



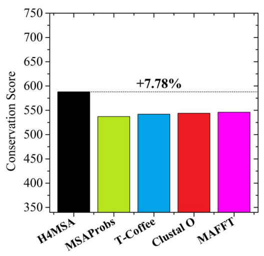
(e) TNF (551)



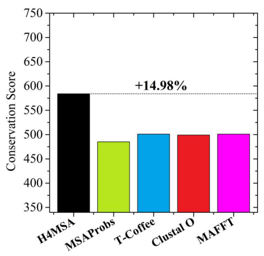
(f) profilin (682)



(g) ricin (740)



(h) trfl (830)



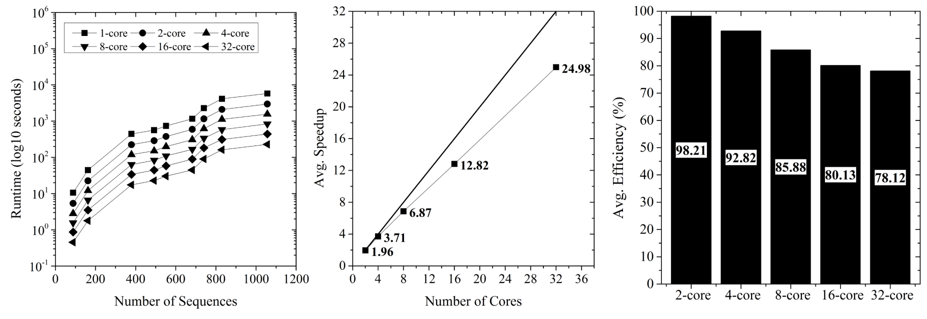
(i) ltn (1056)

Figure 3: Comparison among H4MSA, MSAProbs, T-Coffee, Clustal Ω, and MAFFT in terms of average Conservation Score. Note that, the number of sequences appears between brackets for each dataset.

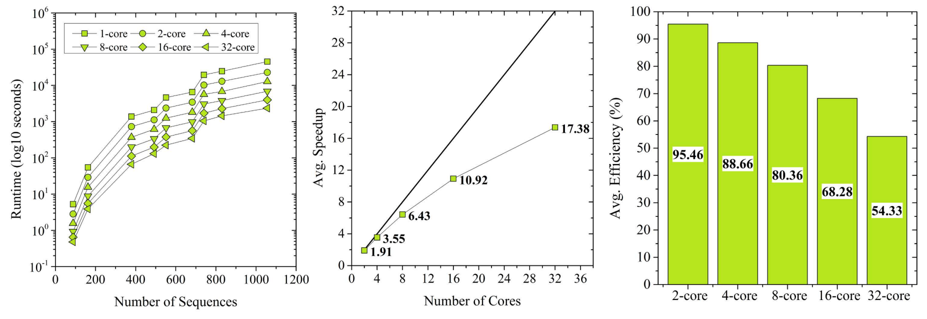
H4MSA is able to obtain accurate alignments.

On the other hand, we compare the parallel performance of the approaches under study. In Table 3 and Table 4, we present the sequential runtime of each method and the parallel speedup and efficiency obtained with different number of cores (2, 4, 8, 16, and 32 cores); respectively. As we can see, the parallel performance of Clustal Ω, T-Coffee, and MAFFT is very poor when the number of cores increases. We can also observe that MSAProbs presents a nice parallel performance with 2, 4, and 8 cores, but its efficiency decreases with 16 and 32 cores. The parallel efficiency of H4MSA remains over 75% in all cases. In Figure 4 and 5, we present a comparison among the five parallel approaches in terms of runtime, speedup, and efficiency.

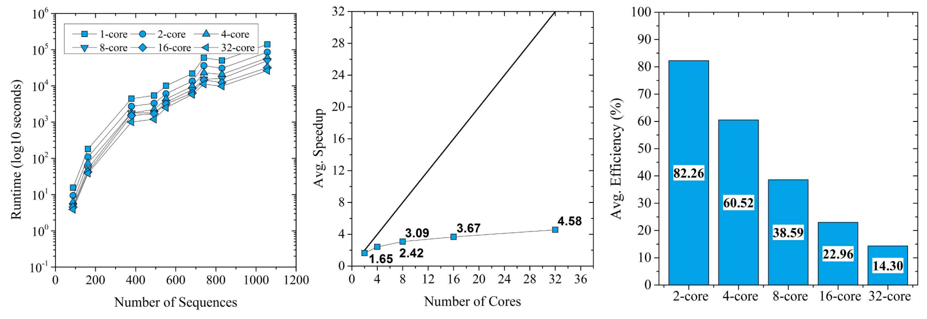
In Figure 4 and 5, if we compare the sequential runtime of each approach, we can see that the fastest algorithms are (in order): Clustal Ω, MAFFT, H4MSA, MSAProbs, and T-Coffee. However, thanks to the parallel efficiency of H4MSA, it is able to be the second fastest approach when the number of cores is set to 32.



(a) H4MSA

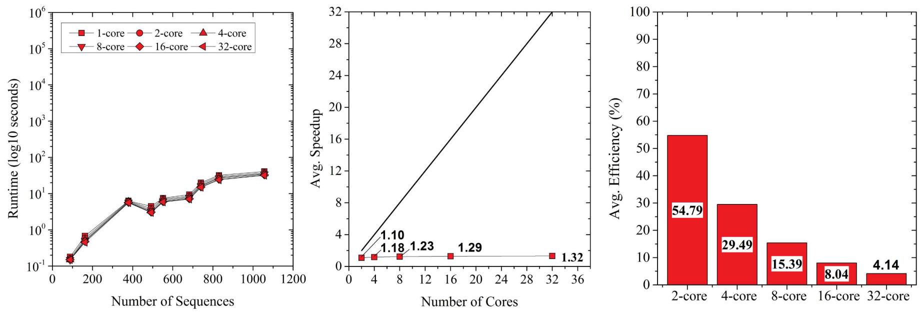


(b) MSAProbs

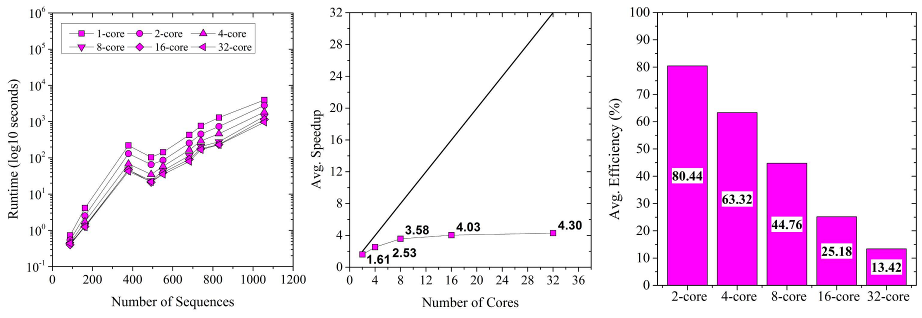


(c) T-Coffee

Figure 4: Runtime, average speedup, and average efficiency (%) obtained by each MSA tool.



(a) Clustal Ω



(b) MAFFT

Figure 5: Runtime, average speedup, and average efficiency (%) obtained by each MSA tool.

All in all, we can conclude that H4MSA is not only an accurate alignment method but also its parallel performance allows it to handle datasets with hundreds of sequences in a reasonable amount of time.

5 Conclusions and Future Works

A parallelization of the Hybrid Multiobjective Memetic Metaheuristic for Multiple Sequence Alignment (H4MSA) is presented in this work. H4MSA is based on the Shuffled Frog-Leaping Algorithm, which provides the benefits of information mixture of the 'shuffled complex evolution' technique. The parallel version of H4MSA has been compared with the parallel approaches of MSAProbs, T-Coffee, Clustal Ω, and MAFFT when solving datasets with different number of sequences in the range [88–1056]. We can conclude that the alignment accuracy and parallel performance of H4MSA is significantly better than other approaches published in the literature.

As future work, we intend to develop a parallel version of H4MSA for shared- and distributed-memory architectures, in order to solve larger datasets.

Acknowledgements

Álvaro Rubio-Largo is supported by the post-doctoral fellowship SFRH/BPD/100872/2014 granted by Fundação para a Ciência e a Tecnologia (FCT), Portugal.

References

[1] D. J. Bacon and W. F. Anderson. Multiple sequence alignment. *J. Mol. Biol.*, 191:153–161, 1986.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2000.

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

[4] R. Doolittle. Similar amino acid sequences: chance or common ancestry? *Science*, 214:149–159, 1981.

[5] R. C. Edgar. MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, 32:1792–1797, 2004.

[6] M. Eusuff, K. Lansey, and F. Pasha. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering Optimization*, 38(2):129–154, 2006.

[7] D. Feng and R. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evolut.*, 25:351–360, 1987.

[8] D. L. Gonzalez-Alvarez, M. A. Vega-Rodriguez, and A. Rubio-Largo. Finding patterns in protein sequences by using a hybrid multiobjective teaching learning based optimization algorithm. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 12(3):656–666, May 2015.

[9] Sandeep K. Gupta, John D. Kececioglu, and Alejandro A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2(3):459–472, 1995.

[10] 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, 2002.

[11] T. Lassmann, O. Frings, and E. L. L. Sonnhammer. Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features. *Nucleic Acids Research*, 37(3):858–865, 2009.

[12] 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.

[13] A. Loytynoja and N. Goldman. An algorithm for progressive multiple alignment of sequences with insertions. *Proceedings of the National Academy of Sciences of the United States of America*, 102(30):10557–10562, 2005.

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

[15] C. Notredame, D. G. Higgins, and J. Heringa. T-Coffee: a novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, 302(1):205 – 217, 2000.

[16] A Rubio-Largo, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez. A multiobjective approach based on artificial bee colony for the static routing and wavelength assignment problem. *Soft Computing*, 17(2):199–211, 2013.

[17] A. Rubio-Largo, M. A. Vega-Rodriguez, and D. L. Gonzalez-Alvarez. Applying moeas to solve the static routing and wavelength assignment problem in optical {WDM} networks. *Engineering Applications of Artificial Intelligence*, 26(5–6):1602 – 1619, 2013.

[18] A. Rubio-Largo, M. A. Vega-Rodriguez, and D. L. Gonzalez-Alvarez. A hybrid multiobjective memetic metaheuristic for multiple sequence alignment. *IEEE Transactions on Evolutionary Computation*, 20(4):499–514, 2016.

[19] Fabian Sievers, Andreas Wilm, David Dineen, Toby Gibson, Kevin Karplus, Weizhong Li, Rodrigo Lopez, Hamish McWilliam, Michael Remmert, Johannes Soding, Julie Thompson, and Desmond Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega. *Molecular Systems Biology*, 7:539, 2011.

[20] J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.

[21] J. D. Thompson, P. Koehl, and O. Poch. BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins*, 61:127–136, 2005.

[22] M.S. Waterman, T.F. Smith, and W.A. Beyer. Some biological sequence metrics. *Advances in Mathematics*, 20(3):367–387, 1976.