A parallel multiobjective metaheuristic for multiple sequence alignment

Alvaro Rubio-Largo\textsuperscript{1} arl@unex.es (+34)927257263
Mauro Castelli\textsuperscript{1} mcastelli@novaims.unl.pt (+351)213 828 610
Leonardo Vanneschi\textsuperscript{1} lvanneschi@novaims.unl.pt (+351)213 828 610
Miguel A. Vega-Rodríguez\textsuperscript{2} mavega@unex.es (+34)927257263

\textsuperscript{1} NOVA IMS, Universidade Nova de Lisboa, 1070-312, Lisbon, Portugal.

\textsuperscript{2} Department of Computer and Communications Technologies, University of Extremadura, Spain.

This is the author accepted manuscript of the following article:


Doi: \url{https://doi.org/10.1101/103101}, \url{https://doi.org/10.1089/cmb.2018.0031}

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License.
A parallel multiobjective metaheuristic for multiple sequence alignment

Authors:

Álvaro Rubio-Largo¹
Mauro Castelli¹,*
Leonardo Vanneschi¹
Miguel A. Vega-Rodríguez²

E-Mail Telephone
arl@unex.es (+34)927257263
mcastelli@novaims.unl.pt (+351)213 828 610
lvanneschi@novaims.unl.pt (+351)213 828 610
mvega@unex.es (+34)927257263

Addresses:

¹ NOVA IMS,
Universidade Nova de Lisboa,
1070-312, Lisbon, Portugal.

² Department of Computer and Communications Technologies,
University of Extremadura,
Spain.

* Corresponding author
Abstract

Background. 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.

Methods. 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.

Results. 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.

Discussion. The comparative study reveals that our parallel aligner obtains better results than MSAProbs, T-Coffee, Clustal Ω and MAFFT. In addition, the parallel version is around 25 times faster than the sequential version with 32 cores, obtaining an efficiency around 80%.

Keywords: Multiple Sequence Alignment, Memetic Computing, Metaheuristic, Multi-Objective Optimization

1 Background

Multiple Sequence Alignment (MSA) is the process of aligning three or more nucleotides/amino-acids sequences at the same time [Bacon and Anderson(1986)].
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 [Doolittle(1981)] [Feng and Doolittle(1987)]. Another important goal of MSA is the determination of biological significance among the given sequences [Notredame(2002)], 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 [Dogan and Otu(2014)]; therefore, different heuristic approaches have been developed for solving this problem in an 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 [Thompson et al.(1994)], Clustal Ω [Sievers et al.(2011)], PRANK [Loytynoja and Goldman(2005)], and Kalign [Lassmann et al.(2009)]. 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) [Notredame et al.(2000)], PROBabilistic CONSistency-based multiple sequence alignment (ProbCons) [Do et al.(2005)], and MSAProbs [Liu et al.(2010)].

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) [Edgar(2004)] and Multiple Alignment using Fast Fourier Transform (MAFFT) [Katoh et al.(2002)].

Multiobjective Optimization in conjunction with Evolutionary Computation is a powerful methodology to optimize real-world problems in the Bioinformatics [Gonzalez-Alvarez et al.(2015)] and Telecommunication fields [Rubio-Largo et al.(2013a)], [Rubio-Largo et al.(2013b)]. In [Rubio-Largo et al.(2016)], we propose a Hybrid Multiobjective Memetic Metaheuristic for the Multiple Sequence Alignment (H4MSA) for optimizing the quality and consistency of the final alignment simultaneously. The results reveal that H4MSA is a very promising tool.

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 $O(k^2 L^k)$ [Waterman et al. (1976)]; 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 $\Omega$, 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 disadvantage of H4MSA is its prohibitive running time when the number of sequences increases. The main contribution of this paper is an efficient parallel version of H4MSA for aligning sets of sequences with a large number of sequences accurately in a reasonable amount of time.

1.1 Multiple Sequence Alignment Problem

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

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 $S$ ($\Sigma$) 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 $\Sigma$, in other words, a column with
all gaps is not allowed. An example of multiple sequence alignment may be:

- **Unaligned Sequences (input):**

  \[ s_1: \text{TSREETK}KHP\text{DASVN}FSEFSK\text{KC}SER\text{K}T\text{M}A\text{KEG}KFEDMA \quad (43) \]
  \[ s_2: \text{RKT}A\text{LENPR}E\text{RNSEIS}KQL\text{GYQWKM}L\text{TEAEKWP}FF\text{QEAQK}L\text{QAM}H \quad (45) \]
  \[ s_3: \text{FM}E\text{KTA}K\text{YAKL}H\text{PEMSNDLTKILSKK}K\text{WKE}L\text{PEKKK}K\text{KYIQDFQEKEQEF} \quad (51) \]
  \[ s_4: \text{VVAESTLK}E\text{S}A\text{INK}\text{K}L\text{GRWRHALSREEE}K\text{YELARKERQLHMQL} \quad (46) \]
  \[ s_5: \text{D}Y\text{KKETE}SDI\text{DKHLS}D\text{ITVVKGSWVA}LGFSDGQEA \quad (36) \]

- **Aligned Sequences (output):**

  \[ s'_1: \text{TSREETK}---KKHP\text{DASVN}FSEFSK\text{KC}SER\text{K}T\text{M}A\text{KEG}KFEDMA---------- \quad (56) \]
  \[ s'_2: \text{R}K---TA---\text{LENPR}E\text{RNSEIS}KQL\text{GYQWKM}L\text{TEAEKWP}FF\text{QEAQK}L\text{QAM}H--- \quad (56) \]
  \[ s'_3: \text{FM}E\text{KTA}K\text{YAKL}H--\text{MEMSDLTKILSKK}K\text{WKE}L\text{PEKKK}K\text{KYIQDFQEKEQEF}--- \quad (56) \]
  \[ s'_4: \text{VVAESTLK}---K------\text{ESS}A\text{INK}\text{K}L\text{GRWRHALSREEE}K\text{YELARKERQLHMQL} \quad (56) \]
  \[ s'_5: \text{D}Y\text{KKETE}----------------\text{SDI}D\text{DKHLS}D\text{ITVVKGSWVA}LGFSDGQEA-------- \quad (56) \]

  \[ * \quad * \quad * \quad * \]

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, \(f_1\)) [Gupta et al. (1995)] and the number of Totally Conserved (\(f_2\)) columns score [Edgar (2004)], [Thompson et al. (2005)].

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

6
\[ WSP(S') = \sum_{l=1}^{AL} SP(l) - \sum_{i=1}^{k} AGP(s'_i) \] (1)

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

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

\[ AGP(s'_i) = (g_o \times \#gaps) + (g_e \times \#spaces) \] (4)

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 is shown in equation 2. Note that in equation 2, \( \delta \) 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 equation 3, where \( LD \) refers to the Levenshtein Distance 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 \( s'_i \) (see equation 4. In equation 4, \( g_o \) 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. In the aforementioned example, the symbol '*' indicates a column
completely aligned with exactly the same compound, therefore, \( f_2 = 4 \).

The maximum number of columns (alignment length) was limited to \( \left\lceil 1.5 \cdot \max(|s_1|, |s_2|, \ldots, |s_k|) \right\rceil \). 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.

The rest of the paper is organized as follows. A detailed description of the parallel H4MSA algorithm is presented in Section 2. Section 3 is devoted to compare the alignment accuracy and parallel efficiency of H4MSA with other parallel approaches published in the literature. Section 4 discusses the obtained results, while Section 5 concludes the paper and suggests future research avenues.

2 Methods

This section is divided into two parts. The first one is devoted to describe the H4MSA algorithm and study the points in which the algorithm spend more time. The second part presents a detailed description about the parallelization process of the metaheuristics.

2.1 H4MSA

The Shuffled Frog-Leaping optimization Algorithm (SFLA) is a memetic metaheuristic developed by Eusuff and Lansey [Eusuff et al. (2006)]. The search procedure begins with a random population of frogs (solutions) in a swamp.

The population of frogs is divided into isolated communities (meme-
plexes) 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 1.1 is:

$s'_1$: 0000000111100000000000000000000000000000000001111111111

$s'_2$: 00111001110000111000000000000000000000000000000000000111

$s'_3$: 00000000000000011000000000000000000000000000000000000111

$s'_4$: 00000000000000000000000000000000000000000000000000000000

$s'_5$: 00000000000000000000000000000000000000000000000000000000

However, 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:

$s'_1$: 2, (8, -2), (47, -9)

$s'_2$: 4, (3, -2), (8, -2), (16, -1), (55, -2)

$s'_3$: 2, (16, -1), (55, -2)
Given a set of \( k \) unaligned sequences \( \mathcal{S} \), \( m \) 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 [Deb et al. (2000)].

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:

\[
\text{TSREETK}---\text{KKHPDA5VNFEFSKCCER0KMTMASEKXGKFED}----------- (56)
\text{RK}-----\text{TA}----\text{LENPR}---\text{MRNSEEISKOLGYKWMLTEAEKRPFF0}----(56)
\text{FMEKTKAKILIP}---\text{MSNSSLTKILSSKKEPEKMKYI}----(56)
\text{VAAESTL}---\text{----------------}--\text{ESAIINKLGRHIALSREEOKYYELEKQRKHEWL} (56)
\text{DYKKETE}-----------\text{SDIKHLSDIHTVWKGSVALGSFD}DQEA-------- (56)
\]

and the following portion of the local worst \( (X_{lw}) \):
and the resultant new frog \( (X_{\text{new}}) \) is constructed by taking into account both portions, filling with new gaps (represented by dots '.') in the example) until obtaining identical length:

\[
\begin{align*}
\text{TS} -& - \text{REETK} - - \text{KHPDASVNFKFXKCSER} - - \text{MKTMASAK} - - \text{EGKDF} - - \text{E-DMA} - - \text{(65)} \\
\text{R} -& - \text{K} - - \text{TA} - - \text{LMNR}-\text{MNSEISQKOL} - - \text{GYNWMLTEAKW} - - \text{PPFQEOAKHLOAMH} - - \text{(65)} \\
\text{FF} -& - \text{K} - - \text{KTMAYKLHPE} - - \text{MSII} - - \text{SOLTILKSIKHELPEKKKM1YIDQFOREKINGF} - - \text{(65)} \\
\text{VVD} -& - \text{AESILK} - - \text{ESSAINK} - - \text{I} - - \text{ILGRMLWSREEEAQ} - - \text{KYLE} - - \text{AEKL} - - \text{ERKEL} - - \text{MOL} - - \text{(65)} \\
\text{DDYKETE} -& - \text{S} - - \text{SIDOHLSGD} - - \text{TVN} - - \text{G} - - \text{SNWALGF} - - \text{SDG} - - \text{GEO} - - \text{(65)}
\end{align*}
\]

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

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

\[
\begin{align*}
\text{TS} -& - \text{REETK} - - \text{KHPDASVNFKFXKCSER} - - \text{MKTMASAK} - - \text{EGKDF} - - \text{E-DMA} - - \text{(70)} \\
\text{R} -& - \text{K} - - \text{TA} - - \text{LMNR}-\text{MNSEISQKOL} - - \text{GYNWMLTEAKW} - - \text{PPFQEOAKHLOAMH} - - \text{(70)} \\
\text{FF} -& - \text{K} - - \text{KTMAYKLHPE} - - \text{MSII} - - \text{SOLTILKSIKHELPEKKKM1YIDQFOREKINGF} - - \text{(70)} \\
\text{VVD} -& - \text{AESILK} - - \text{ESSAINK} - - \text{I} - - \text{ILGRMLWSREEEAQ} - - \text{KYLE} - - \text{AEKL} - - \text{ERKEL} - - \text{MOL} - - \text{(70)} \\
\text{DDYKETE} -& - \text{S} - - \text{SIDOHLSGD} - - \text{TVN} - - \text{G} - - \text{SNWALGF} - - \text{SDG} - - \text{GEO} - - \text{(70)}
\end{align*}
\]

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

\[
\begin{align*}
\text{TS} -& - \text{REETK} - - \text{KHPDASVNFKFXKCSER} - - \text{MKTMASAK} - - \text{EGKDF} - - \text{E-DMA} - - \text{(70)} \\
\text{R} -& - \text{K} - - \text{TA} - - \text{LMNR}-\text{MNSEISQKOL} - - \text{GYNWMLTEAKW} - - \text{PPFQEOAKHLOAMH} - - \text{(70)} \\
\text{FF} -& - \text{K} - - \text{KTMAYKLHPE} - - \text{MSII} - - \text{SOLTILKSIKHELPEKKKM1YIDQFOREKINGF} - - \text{(70)} \\
\text{VVD} -& - \text{AESILK} - - \text{ESSAINK} - - \text{I} - - \text{ILGRMLWSREEEAQ} - - \text{KYLE} - - \text{AEKL} - - \text{ERKEL} - - \text{MOL} - - \text{(70)} \\
\text{DDYKETE} -& - \text{S} - - \text{SIDOHLSGD} - - \text{TVN} - - \text{G} - - \text{SNWALGF} - - \text{SDG} - - \text{GEO} - - \text{(70)}
\end{align*}
\]

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.

\[
\begin{align*}
\text{TSR} & \quad \text{EETK} \quad \text{KKHPASVNFSEFSSKCCSER} \quad \text{WKMTSAK} \quad \text{EKGF} \quad \text{EDMA} \quad \text{(78)} \\
\text{RK} & \quad \text{TA} \quad \text{LENMPRINSEIKSL} \quad \text{GYDVMKLTAEDK} \quad \text{FFDOE} \quad \text{AKLOAMH} \quad \text{(78)} \\
\text{FMP} & \quad \text{ETKAYAKLHE} \quad \text{M} \quad \text{SOLTKILSKKKKPEKKNNYIDQ} \quad \text{FOREKGF} \quad \text{(78)} \\
\text{VYA} & \quad \text{ESTLK} \quad \text{ESSAIK} \quad \text{ILGRNHALSREEQA} \quad \text{KYEL} \quad \text{ARKERQIMLQ} \quad \text{(78)} \\
\text{D} & \quad \text{YKKETE} \quad \text{DIDKHLSDIN} \quad \text{TVNK} \quad \text{G5WALGFSDQ} \quad \text{GOEA} \quad \text{(78)} \quad \text{←}
\end{align*}
\]

\[
\begin{align*}
\text{TSR} & \quad \text{EETK} \quad \text{KKHPASVNFSEFSSKCCSER} \quad \text{WKMTSAK} \quad \text{EKGF} \quad \text{EDMA} \quad \text{(78)} \\
\text{RK} & \quad \text{TA} \quad \text{LENMPRINSEIKSL} \quad \text{GYDVMKLTAEDK} \quad \text{FFDOE} \quad \text{AKLOAMH} \quad \text{(78)} \\
\text{FMP} & \quad \text{ETKAYAKLHE} \quad \text{M} \quad \text{SOLTKILSKKKKPEKKNNYIDQ} \quad \text{FOREKGF} \quad \text{(78)} \\
\text{VYA} & \quad \text{ESTLK} \quad \text{ESSAIK} \quad \text{ILGRNHALSREEQA} \quad \text{KYEL} \quad \text{ARKERQIMLQ} \quad \text{(78)} \\
\text{D} & \quad \text{YKKETE} \quad \text{DIDKHLSDIN} \quad \text{TVNK} \quad \text{G5WALGFSDQ} \quad \text{GOEA} \quad \text{(78)} \quad \text{←}
\end{align*}
\]

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

\[
\begin{align*}
\text{TSR} & \quad \text{EETK} \quad \text{KKHPASVNFSEFSSKCCSER} \quad \text{WKMTSAK} \quad \text{EKGF} \quad \text{EDMA} \quad \text{(78)} \\
\text{RK} & \quad \text{TA} \quad \text{LENMPRINSEIKSL} \quad \text{GYDVMKLTAEDK} \quad \text{FFDOE} \quad \text{AKLOAMH} \quad \text{(78)} \\
\text{FMP} & \quad \text{ETKAYAKLHE} \quad \text{M} \quad \text{SOLTKILSKKKKPEKKNNYIDQ} \quad \text{FOREKGF} \quad \text{(78)} \\
\text{VYA} & \quad \text{ESTLK} \quad \text{ESSAIK} \quad \text{ILGRNHALSREEQA} \quad \text{KYEL} \quad \text{ARKERQIMLQ} \quad \text{(78)} \\
\text{D} & \quad \text{YKKETE} \quad \text{DIDKHLSDIN} \quad \text{TVNK} \quad \text{G5WALGFSDQ} \quad \text{GOEA} \quad \text{(78)} \quad \text{←}
\end{align*}
\]

(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 [Lassmann et al. (2009)] 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, (ii) remove all gaps in the selected portion, (iii) re-align the portion with Kalign2 [Lassmann et al. (2009)] method, and (iv) replace the old portion by the new one. In the following we can see an example:

(j) Replace $X_{lw}$ by $X_{new}$.

(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 ($f_2$).

2.2 Parallel Scheme

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. In Figure 1, we present a flowchart of the parallel scheme of H4MSA.

As we can see, 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). We recommend \( M=m \), that is to say, \( m \) threads in charge of generating \( n \) random frogs in parallel. Following this advice, we obtain the best parallel performance of H4MSA.

The second and third steps of H4MSA (Fast Non-Dominated Sorting procedure and the shuffle procedure) are carried out in a single-thread mode, that is to say, only one thread is in charge of carrying out these two steps. The main reason is because the sorting and division process consumes no significant runtime in comparison with the other tasks of H4MSA. These steps consume \(~0.1\%\) of a single generation runtime.

In Step 4, H4MSA performs \( m \) independent evolutionary processes of \( N \) iterations. It is clear that the tasks carried out within each evolutionary process (learning process, mutation process, and local search) are computationally expensive. Therefore, each of the \( m \) threads are in charge of evolving a memeplex in parallel. The workload of each memeplex may be different for each thread; therefore, instead of using a static distribution where each of the \( m \) thread is in charge of performing the consecutive \( \frac{N}{m} \) iterations of the evolutionary process, in this parallel version, we use a dynamic distribution scheduling of the
iterations among the threads during runtime.

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.

3 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 multicore environments:

- **MSAProbs** (version 0.9.7) [Liu et al.(2010)]. 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) [Notredame et al.(2000)]. 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) [Sievers et al.(2011)]. It is the latest addition to the Clustal family. It offers a significant increase in scalability over previous versions, allowing a large number of sequences to be aligned. It also makes use of multiple processors by using the --threads flag.

• **MAFFT** (version 7.215) [Katoh et al.(2002)]. 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: \( m=M \) (number of memeplexes equal to number of threads), \( n=32 \) (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 [Sievers et al.(2011)]. 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.

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 cluster with two AMD Opteron™ Processor 6376 of 16 cores (a total of 32 cores) at 2.3GHz, 12MB Cache L3, 48 GB of DDR3 RAM, and Scientific Linux 6.1.

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 ($M$).

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.
4 Discussion

This section discusses the obtained results and allows to draw important considerations with respect to the proposed system.

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 H4MSA is able to obtain accurate alignments.

To ensure that the differences among the approaches in our comparative study are statistically significant, we perform a Wilcoxon signed-rank test between each pair of methods by using a confidence level of 1% (p-value < 0.01). As we can see in Table 3, we can conclude that the differences of Conservation Score between H4MSA and any other method result statistically significant in the nine benchmarks tested.

On the other hand, we compare the parallel performance of the approaches under study. In Table 4, we present the parallel speedup and efficiency obtained with different number of cores (M = 2, 4, 8, 16, and 32 cores). 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 cores.
and 32 cores. The parallel efficiency of H4MSA remains over 75% in all cases. In Figure 4, we present a comparison among the five parallel approaches in terms of runtime, speedup, and efficiency.

In Figure 4, 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.

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

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, with the aim of solving larger datasets, we intend to develop a parallel version of H4MSA for shared- and distributed-
memory architectures. In addition, we intend to develop other parallel
swarm intelligence metaheuristics, e.g. [Rubio-Largo et al.(2012)].

Acknowledgements

This work was partially funded by the AEI (State Research Agency,
Spain) and the ERDF (European Regional Development Fund, EU),
under the contract TIN2016-76259-P (PROTEIN project). 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.

Author Disclosure Statement

No competing financial interests exist.
References


Table 1: Selected HOMFAM datasets

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

Table 2: Average conservation score in 30 independent runs

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

Table 3: Sets of algorithms where the differences of Conservation Score are statistically not significant (p-value ≥ 0.01).

Seatoxin (MSAProbs, T-Coffee, MAFFT)
Hip (T-Coffee, MAFFT)
Cyt3 -
Rnasemam (MSAProbs, Clustal Ω) and (T-Coffee, MAFFT)
TNF (MSAProbs, Clustal Ω) and (T-Coffee, MAFFT)
Profilin -
Ricin -
Trfl (Clustal Ω, MAFFT)
Ltn (T-Coffee, Clustal Ω, MAFFT)
Table 4: Speedup and Efficiency (%) obtained by the multi-threaded MSA approaches.

<table>
<thead>
<tr>
<th>#Cores</th>
<th>H4MSA</th>
<th>MSAProbs</th>
<th>T-Coffee</th>
<th>Clustal Ω</th>
<th>MAFFT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>E</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>1.96</td>
<td>98.21</td>
<td>1.91</td>
<td>95.46</td>
<td>1.65</td>
</tr>
<tr>
<td>4</td>
<td>3.71</td>
<td>92.82</td>
<td>3.55</td>
<td>88.66</td>
<td>2.42</td>
</tr>
<tr>
<td>8</td>
<td>6.87</td>
<td>85.88</td>
<td>6.43</td>
<td>80.36</td>
<td>3.09</td>
</tr>
<tr>
<td>16</td>
<td>12.82</td>
<td>80.13</td>
<td>10.92</td>
<td>68.28</td>
<td>3.67</td>
</tr>
<tr>
<td>32</td>
<td>24.98</td>
<td>78.12</td>
<td>17.38</td>
<td>54.33</td>
<td>4.58</td>
</tr>
</tbody>
</table>

S: Average speedup in the nine datasets of HOMFAM.
E: Average efficiency (%) in the nine datasets of HOMFAM.
A total of $n \times m$ random frogs are generated in parallel and distribution among the $m$ memeplexes.

Determine $X_{lb}$, $X_{lw}$ and $X_{gb}$.

Learning and Mutation Process ($X_{lb}$)

Is $X_{new}$ better than $X_{lw}$?

Learning and Mutation Process ($X_{gb}$)

Is $X_{new}$ better than $X_{lw}$?

Local Search (Kalign2)

$X_{lw} = X_{new}$

Stopping Criterion?

Yes

Output set of non-dominated solutions

No

Generate random frog

Yes

No

Mix the $m$ memeplexes for the next generation

$N$ generations reached?

Yes

No

Figure 1: Parallel flowchart of H4MSA
Figure 2: Examples of Static vs. dynamic distribution of iterations.
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.
Figure 4: Runtime, average speedup, and average efficiency (%) obtained by each MSA tool.