Swarm intelligence for optimizing the parameters of multiple sequence aligners

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

[a] NOVA IMS, Universidade Nova de Lisboa, 1070-312, Lisboa, Portugal

[b] Depto. of Computer and Communications Technologies, University of Extremadura, 10003, Cáceres, Spain

This is the accepted author *manuscript of the following article published by Elsevier*:

Rubio-Largo, Á., Vanneschi, L., Castelli, M., & Vega-Rodríguez, M. A. (2018). Swarm intelligence for optimizing the parameters of multiple sequence aligners. Swarm and Evolutionary Computation, 42, 16-28. DOI: 10.1016/j.swevo.2018.04.003

Swarm Intelligence for Optimizing the Parameters of Multiple Sequence Aligners

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

[a] *NOVA IMS, Universidade Nova de Lisboa, 1070-312, Lisboa, Portugal*
[b] *Depto. of Computer and Communications Technologies, University of Extremadura, 10003, Cáceres, Spain*

Abstract

Different aligner heuristics can be found in the literature to solve the Multiple Sequence Alignment problem. These aligners rely on the parameter configuration proposed by their authors (also known as default parameter configuration), that tried to obtain good results (alignments with high accuracy and conservation) for any input set of unaligned sequences. However, the default parameter configuration is not always the best parameter configuration for every input set; namely, depending on the biological characteristics of the input set, one may be able to find a better parameter configuration that outputs a more accurate and conservative alignment. This work's main contributions include: to study the input set's biological characteristics and to then apply the best parameter configuration found depending on those characteristics. The framework uses a pre-computed file to take the best parameter configuration found for a dataset with similar biological characteristics. In order to create this file, we use a Particle Swarm Optimization (PSO) algorithm, that is, an algorithm based on swarm intelligence. To test the effectiveness of the characteristic-based framework, we employ five well-known aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. The results of these aligners see clear improvements when using the proposed characteristic-based framework.

Keywords: Swarm intelligence, Multiple sequence alignment, characteristics-based framework, evolutionary algorithms

1. Introduction

The simultaneous alignment of three or more Nucleotides/ Amino-Acids (AA) sequences is known in the literature as the Multiple Sequence Alignment (MSA) problem [1], and is considered an NP-complete optimization problem [10]. The MSA problem can be defined as follows:

Email addresses: arl@unex.es (Álvaro Rubio-Largo), lvanneschi@novaims.unl.pt (Leonardo Vanneschi), mcastelli@novaims.unl.pt (Mauro Castelli), mavega@unex.es (Miguel A. Vega-Rodríguez)

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 Σ, (for example the AA or the nucleotides alphabets), a MSA of S is defined as the set S': $\{s'_1, s'_2, \ldots, s'_k\}$, where the length of all the k sequences is exactly the same. Note that, S' is defined over the same alphabet as S (Σ) with an additional gap symbol ($-$); S' is thus 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, (namely, a column with all gaps is not allowed). According to [10], the complexity of finding an optimal alignment is $O(k2^k L^k)$, where k is the number of sequences and L is the $max(|s_1|, |s_2|, \ldots, |s_k|)$. In the following, we present an example of MSA:

Unaligned set (S):				Aligned set (S'):		
s_1:	`GDNI`	(4)		s'_1:	`GDNI--`	(6)
s_2:	`KQLTQD`	(6)	$\rightarrow$	s'_2:	`KQLTQD`	(6)
s_3:	`ACRKN`	(5)		s'_3:	`ACRK-N`	(6)

A well-conserved alignment leads to extra biological significance [31]; therefore, MSA is frequently employed to produce strong biological facts about proteins. Further, MSA mainly focuses on reflecting biological relationships among different sequences, which is an essential step for inferring phylogenetic relationships [11], [16]. Another important feature of an accurate MSA is that it allows the determination of genes that are susceptible to mutation.

In the literature, we find a range of approaches to deal with the MSA problem. While almost all of them allow us to modify some specific parameters by using different flags, if no flags are used then a default configuration is used. The default configuration is proposed by the developers of the aligner and refers to the best parameter configuration found for aligning any input set of unaligned sequences with a reasonable level of accuracy and conservation. However, the default configuration may not always be the best choice for every input set. Depending on the biological characteristics of the input set, a better configuration may be used to obtain a more accurate and conservative alignment. This is the idea of the proposed characteristic-based framework: to study the biological characteristics of the input set and, consequently, to apply a certain configuration depending on those characteristics. Therefore, the characteristic-based framework uses three input files: aligner (executable), a set of unaligned sequences, and a *characteristics-configuration file*. Note that the *characteristics-configuration file* depends on the input aligner, and contains the best parameter configuration of the aligner for a number of biological datasets with different characteristics. A swarm intelligence approach is applied to optimize the parameters of an input aligner, thereby obtaining its *characteristics-configuration file*. In this way, the framework will run the aligner with the best parameter configuration found for another dataset with similar biological characteristics, improving the input aligner's accuracy and conservation. In [37], we present a preliminary version of the framework.

As demonstrated by a series of recent publications [19, 20], in compliance with the 5-step rule [7] when developing a really useful sequence-based method for a biological system we should follow these five guidelines: (a) construct or select a valid benchmark dataset to train and test the model; (b) formulate the biological sequence samples with

an effective mathematical expression that can truly reflect their intrinsic correlation with the target to be identified; (c) introduce or develop a powerful algorithm (or engine) to operate the analysis; (d) properly perform cross-validation tests to objectively evaluate the anticipated accuracy of the model; and (e) establish a user-friendly web-server for the analysis method that is accessible to the public. Below, we describe how to deal with these steps individually.

The biggest contributions of this work include the following: a characteristic-based framework for improving the quality alignment of any aligner, diverse biological characteristics for describing a set of unaligned sequences, and a comparative study on the framework's effectiveness when it is applied to five aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE.

The rest of the paper is organized as follows. A description of related works is provided in Section 2. In Section 3, we detail the characteristic-based framework. Section 4 compares the framework's accuracy with other aligners published in the literature. Finally, in Section 5 we summarize the conclusions extracted from the study and describe some avenues for future work.

2. Related Work

Traditionally, exact approaches for MSA, such as dynamic programming, have been used. Yet these methods become computationally prohibitive when the number of sequences increases. In the literature, we find different heuristics for MSA that are categorized in three groups: progressive-based methods, consistency-based methods, and iterative refinement methods.

In the first group, we find the progressive-based methods, which are widely used [18]. Basically, given a set of unaligned sequences a progressive-based method computes a distance matrix from every pair of sequences. After that, it employs a hierarchical clustering algorithm, such as the Unweighted Pair Group Method with Arithmetic Mean (UPGMA) or Neighbor-Joining (NJ), in order to build a guide-tree. The last step is to perform the alignment among the given sequences by following the guide-tree. Several progressive-based methods exist, such as Clustal W [42], PRANK [28], Fast Statistical Alignment (FSA) [4], Kalign [24], and DIALIGN-TX [41].

In the second group, we have the consistency-based methods. These approaches build a database with the local and global alignments between every pair of sequences. According to [12], the consistency-based approaches first harness the information contained within regions that are consistently aligned among a set of pairwise superpositions in order to realign pairs of proteins through both global and local refinement methods. Among the most important consistency-based methods are: Tree-based Consistency Objective Function For alignment Evaluation (T-Coffee) [32], PROBabilistic CONSistency-based multiple sequence alignment (ProbCons) [9], ProbAlign [36], and MSAProbs [26].

In the third group, we find the iterative refinement methods. These focus on correcting an erroneous gap inserted at an early stage of a progressive alignment. The first step in these methods is to build an initial MSA by using any progressive-based method. The second step consists of dividing the guide-tree of the initial alignment into two subtrees which are re-aligned with the aim of obtaining an improved new alignment. The second step is iteratively repeated until a certain number of iterations is reached.

There exist several iterative refinement methods, such as MUltiple Sequence Comparison by Log-Expectation (MUSCLE) [13], Multiple Alignment using Fast Fourier Transform (MAFFT) [21], ProbCons [9] (it allows the option of a final iterative refinement), and MUMMALS [34]. In this group, we can find some evolutionary and/or genetic algorithms techniques for the MSA problem: VDGA [29], GAPAM [30], MO-SAStrE [33], HMOABC [39], H4MSA [40].

To avoid completely losing the sequence-order information, the concept of PseAA (pseudo amino acid) composition was proposed [6]. In contrast with the conventional amino acid composition that contains 20 components with each reflecting the occurrence frequency for one of the 20 native amino acids in a protein, the PseAA composition contains a set of greater than 20 discrete factors, where the first 20 represent the components of its conventional AA composition while the additional factors incorporate some sequence-order information via various modes. Some very powerful bioinformatics tools for analyzing biological sequences were recently developed, e.g. a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition [17] or an effective formulation for analyzing genomic sequences [5]. When harnessing Pse-in-One 2.0 [5], users only need to input DNA, RNA, or protein sequences as well as their selected or defined features, and can immediately obtain the corresponding feature vectors suitable for any of the existing machine-learning programs to conduct various analyses. All the aforementioned works may be considered a starting point for the characteristic-based framework proposed in this paper, which analyzes the composition of the protein sequences in order to select a proper parameter configuration.

3. Characteristic-based Framework

This section is divided into two. The first subsection describes how the *characteristic-configuration* file is obtained, while the second discusses the main properties of the proposed framework.

3.1. Characteristic-configuration File

The characteristic-based framework receives the following inputs: aligner, set of unaligned sequences, and a *characteristic-configuration* file. The *characteristic-configuration* file contains the best parameter configuration of the given aligner for diverse sets of unaligned sequences with different biological characteristics (one per line). The line structure of the *characteristic-configuration* file will be:

```
A1;A2;A3;B1;B2;C1;C2;C3;C4;C5;<best-conf>
```

On one hand, we find three different groups of characteristics (A, B, and C), a total of 10 characteristics. In the following, we present a description of each characteristic within each group:

A1. Number of unaligned sequences.

A2. Average length of the unaligned sequences.

A3. Standard Deviation of the Length.

B1. Average Kimura Distance (evolutionary distance, [23]) between each pair of unaligned sequences.

B2. Standard deviation of the Kimura Distance.

C1. Percentage of AA with electrically charged side chains (Positive): `R`, `H`, and `K`.

C2. Percentage of AA with electrically charged side chains (Negative): `D`, and `E`.

C3. Percentage of AA with polar uncharged side chains: `S`, `T`, `N`, and `Q`.

C4. Percentage of special AA cases: `C`, `U`, `G`, and `P`.

C5. Percentage of AA with hydrophobic side chains: `A`, `V`, `I`, `L`, `M`, `F`, `Y`, and `W`.

The features within group A refers to the number and length of the unaligned sequences. Within group B, we find the features related to the evolutionary distance between each pair of unaligned sequences (average and standard deviation).

According to [27], amino-acids (AA) are the monomeric (single polypeptide chain) building block of proteins. Each amino acid has the same fundamental structure, a central α carbon atom (C_α) of AA, which is adjacent to the carboxyl group and bonded to four different chemical groups: (i) an amino (NH_2) group, (ii) a carboxyl ($COOH$) group, (iii) a hydrogen (H) atom, and (iv) a variable group, called a *side-chain* or R group. Amino acids can be classified into a few distinct categories based primarily on their solubility in water, as influenced by the polarity of their side chains. Amino acids with polar side groups tend to be on the surface of proteins; by interacting with water, they make proteins soluble in aqueous solutions. In contrast, amino acids with non-polar side groups avoid water and aggregate to form the water insoluble core of proteins. Therefore, in group C we find the side-chain features of sequences (in %): weak acid (positive), weak base (negative), polar, special, and hydrophobic.

As seen, a total of 10 characteristics divided into three groups are chosen. In the first group (A), we consider the number of sequences and their average and standard deviation length because these characteristics significantly influence the performance of the aligners, mainly in running time issues. The Evolutionary Distance used within the second group of characteristics (B) has been widely used to validate new multiple sequence aligners in the literature, e.g. [24]. Finally, in group C we consider the composition of the input sequences based on their side-chain characteristics because the polarity of amino acid side chains is one of the forces responsible for shaping the final three-dimensional structure of proteins, [27].

Note that, for sets of DNA sequences, groups of characteristics A and B are valid; however, the group of characteristics C (side-chain features) will be replaced by the percentage of each DNA base (`A`, `C`, `T`, and `G`).

For example, given the set of unaligned sequences of Section 1, we can extract the following characteristics:

- Characteristics A: 3 unaligned sequences (A1), 17.334 of average length (A2) and 5.312 of standard deviation (A3).
- Characteristics B: Average Kimura distance of 0.2344 (B1), standard deviation of 0.0719 (B2).

Algorithm 1: Particle Swarm Optimization

```
1  x_gb.f ← -1 ;
2  foreach x_i^0 in {x_1^0 ... x_N^0} do
3      x_i^0 ← RandomPositionInSearchSpace(LB,UB);
4      x_i^0.f ← EvaluateParameterConfiguration(x_i^0);
5      if (x_i^0.f > x_gb.f) then
6          x_gb ← x_i^0 ;
7  t ← 1 ;
8  repeat
9      foreach x_i in {x_1 ... x_N} do
10         rand1 ← random(0,1);
11         rand2 ← random(0,1);
12         W ← 0.7 ;
13         C1 ← random(0,2);
14         C2 ← 2 - C1 ;
15         x_n ← NeighbourParticleTo(x_i^t) ;
16         for j ← 1 to D do
17             v_{i,j}^{t+1} ← (W × v_{i,j}^t) +
18                     (C1 × rand1 × (x_gb.p_j − x_i^t.p_j)) +
19                     (C2 × rand2 × (x_n.p_j − x_i^t.p_j));
20             x_i^{t+1}.p_j ← x_i^t.p_j + v_{i,j}^{t+1} ;
21         x_i^{t+1}.f ← EvaluateParameterConfiguration(x_i^{t+1});
22         if (x_i^{t+1}.f > x_gb.f) then
23             x_gb ← x_i^{t+1} ;
24     t ← t + 1 ;
25 until Stopping Criterion is satisfied;
26 return x_gb
```

- Characteristics C: 7.52% of positive AA (C1), 41.63% of negative AA (C2), 16.68% of uncharged AA (C3), 5.81% of special AA, and 28.36% of hydrophobic AA.

On the other hand, on each line of the *characteristic-configuration* file, we also find the *best-parameter-configuration* for the given aligner. In order to find the *best parameter configuration* for a given set of unaligned sequences, we implemented a Particle Swarm Optimization algorithm (PSO) [22].

PSO is a population based approach, inspired by social behaviour of bird flocking or fish schooling. In PSO, the potential solutions (*particles*) fly through the search-space following the current optimum particles. The movements of the particles are guided by the best known position of each particle in the search space as well as the entire swarm's best known position. The process is repeated until a satisfactory solution is discovered. A pseudocode of the PSO algorithm is presented in Algorithm 1.

As seen in Algorithm 1, PSO is initialized with a group of random particles (lines 2-6) and then iterates for optima by updating generations. In every iteration (lines 9-25), the

movement of each particle is influenced by its local best known position (line 15), but is also guided toward the best known particle found in the search-space (x_{gb}), which are updated as better positions are found by other particles (lines 22-23). This is expected to move the swarm toward the best solutions. For a complete explanation of PSO, please refer to [22].

In this paper, the input parameters of PSO are: (i) stopping criterion, (ii) set of unaligned sequences, (iii) reference aligned set of sequences or true alignment, (iv) aligner binary program, (v) list of parameters to optimize, and (vi) upper/lower bounds of the parameters.

Each particle consists of three elements: (i) chromosome, (ii) velocity, and (iii) fitness value. The chromosome-encoding in PSO is a list of D parameters $\{p_1, p_2, \ldots, p_D\}$ to optimize. All the parameters have lower and upper bounds that are denoted with LB=$\{lb_1, lb_2, \ldots, lb_D\}$ and UB=$\{ub_1, ub_2, \ldots, ub_D\}$, respectively. After running the aligner with the list of selected parameters, an alignment is obtained (A). In order to quantify the agreement between the true alignment and the alignment A, two measures are used: Q-score and TC-score [13]. Q-score (Quality score) indicates the number of correctly aligned residue pairs divided by the number of residue pairs in the reference alignment. This measure is also known as the Sum-of-Pairs (SP) score. The TC-score (Total Column score) is the number of correctly aligned columns divided by the number of columns in the reference alignment and is also known as the Column Score (CS). Therefore, the goal of the PSO algorithm is to find a parameter-configuration that produces alignment (A) that maximizes the following objective function (f):

$$f(A) = (0.5 * Q) + (0.5 * TC) \tag{1}$$

As mentioned, each particle i stores its current position x_i^t (chromosome) and velocity v_i^t at time t. In addition, PSO keeps track of the global best known particle (x_{gb}), which will be the output of the algorithm (the best parameter configuration found).

As we can see on lines 12-14 of Algorithm 1, the PSO algorithm uses three constants (W, $C1$, and $C2$) to control the three directions which determine the next velocity and position of the particles. In the velocity update of the particle (lines 17-19 in Algorithm 1), three components exist which are usually referred as *inertia* ($W \times v_i^t$), *social influence* ($C1 \times rand1 \times (x_{gb}.p_j - x_i^t.p_j)$), and *neighborhood influence* ($C2 \times rand2 \times (x_n.p_j - x_i^t.p_j)$). These components use an element of randomness, so PSO allows particles to explore novel areas of the search space, avoiding stagnation in local minima.

3.2. Explanation of the characteristic-based framework

As mentioned in Section 1, the characteristic-based framework requires three input elements: aligner, a *characteristic-configuration* file for the given aligner and a set of unaligned sets of sequences. In the following, we describe the steps of the framework:

Step 1. Extract the biological characteristics explained in the previous subsection from the input set of unaligned sequences: characteristics A (A1, A2, A3), characteristics B (B1, B2), and characteristics C (C1, C2, C3, C4, and C5).

Step 2. Obtain the parameter-configuration.

Step 2.1. Normalize (in the 0-1 range) the characteristics A, B, and C of the input set and also the characteristics A, B, and C included in the *characteristic-configuration* file.

Step 2.2. Obtain the closest parameter configuration in terms of characteristics A, B, and C by using the Euclidean Distance; we refer to it as configuration A, configuration B, and configuration C.

Step 3. In Parallel, run the aligner with the parameter-configuration A, B, C, and also with the default parameter configuration. In this way, we use four threads, each one running the same aligner with a different parameter-configuration. Once the alignment is obtained, the thread will compute the sum-of-pairs profile score (VTML240) to determine the quality of the alignment:

```
muscle3.8 -spscore <output-alignment> -sv
```

Step 4. Return the alignment with the highest sum-of-pairs profile score (VTML240).

For example, consider the following *characteristic-configuration* file:

	A1	**A2**	**A3**	**B1**	**B2**	**C1**	**C2**	**C3**	**C4**	**C5**	
1)	46	251.87	125.69	0.2031	0.0529	11.76%	14.26%	18.67%	15.97%	39.35%	`<conf1>`
2)	53	321.47	14.09	0.3991	0.1224	12.36%	12.25%	18.64%	12.69%	44.05%	`<conf2>`
3)	4	118	56.29	0.212	0.024	17.47%	11.86%	21.76%	9.99%	38.93%	`<conf3>`
4)	35	689.8	293.06	0.2834	0.1482	14.37%	12.76%	18.39%	15.46%	39.01%	`<conf4>`
5)	69	490.75	293.72	0.2495	0.1195	13.87%	12.60%	17.58%	15.63%	40.32%	`<conf5>`
6)	4	232	7.97	0.352	0.0524	8.54%	9.05%	23.34%	16.81%	42.27%	`<conf6>`

Given a set of unaligned sequences, the first step is to extract its biological characteristics:

	A1	**A2**	**A3**	**B1**	**B2**	**C1**	**C2**	**C3**	**C4**	**C5**
7)	5	128.2	74.86	0.1960	0.0536	13.34%	12.07%	18.03%	16.36%	40.20%

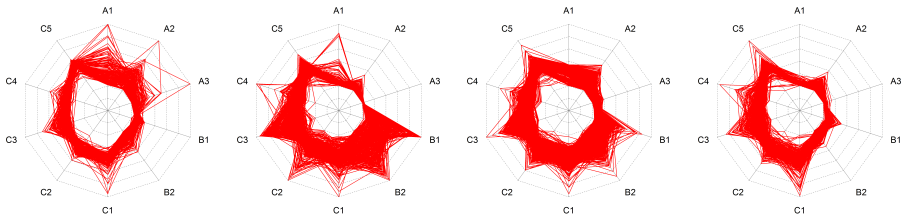
Then, we perform a normalization of the characteristics in the 0-1 range:

	A1	**A2**	**A3**	**B1**	**B2**	**C1**	**C2**	**C3**	**C4**	**C5**	
1)	0.6462	0.2341	0.412	0.0348	0.2323	0.3613	1	0.1881	0.8763	0.083	`<conf1>`
2)	0.7538	0.3558	0.0214	1	0.7922	0.4277	0.6152	0.1841	0.3972	1	`<conf2>`
3)	0	0	0.1691	0.0788	0	1	0.5396	0.7259	0	0	`<conf3>`
4)	0.4769	1	0.9977	0.4303	1	0.6535	0.7122	0.1408	0.8027	0.0172	`<conf4>`
5)	1	0.6519	1	0.2635	0.7688	0.597	0.6815	0	0.8271	0.2725	`<conf5>`
6)	0	0.1994	0	0.7683	0.2284	0	0	1	1	0.6521	`<conf6>`
7)	0.0154	0.0178	0.2341	0	0.2384	0.5373	0.5802	0.0773	0.9348	0.2491	

To obtain the best parameter-configuration A, B, and C, we calculate the Euclidean distance between the characteristics A, B, and C of the input set of unaligned sequences and the characteristics A, B, and C of each line in the *characteristic-configuration* file:

	1)	**2)**	**3)**	**4)**	**5)**	**6)**
Characteristics A (A1, A2, A3)	0.690	0.840	**0.069**	1.327	1.399	0.297
Characteristics B (B1, B2)	**0.035**	1.143	0.251	0.875	0.592	0.768
Characteristics C (C1, C2, C3, C4, C5)	0.501	0.937	1.254	0.326	**0.179**	1.282

As we can see, we have selected `<conf3>` as configuration A, `<conf1>` as configuration B, and `<conf5>` as configuration C. Finally, the last step is to simultaneously run the aligner with the aforementioned parameter-configurations (and also the default) and return the alignment with the highest sum-of-pairs profile score (VTML240).



(a) BAliBASE v3.0 (b) OX-Bench (c) PREFAB v4.0 (d) SABMark v1.65

Figure 1: The chart contains the star plot of the four benchmark suites tested in this work: BAliBASE v3.0 (218 datasets), OX-Bench (395 datasets), PREFAB v4.0 (1680 datasets), and SABMark v1.65 (423 datasets). The variable list for each sample star plot is: group of characteristics A (A1, A2, A3), group of characteristics B (B1, B2), and group of characteristics C (C1, C2, C3, C4, C5).

4. Experimental Results

In this section, we study the influence of the characteristic-based framework on different multiple sequence aligners published in the literature. We explain the methodology followed in the comparative study. Then, we present the optimization of the parameters of each aligner. Finally, we analyze and discuss the results obtained by the characteristic-based framework versions and other aligners published in the literature.

4.1. Methodology

In order to prove the effectiveness and accuracy of the characteristic-based framework, we selected the following well-known aligners: Clustal W [42] (v2.1), DIALIGN-TX [41] (v1.0.2), Kalign2 [24] (v2.03), MAFFT [21] (v7.215), and MUSCLE [13] (v3.8). These aligners were selected because they employ several parameters to optimize and present a reasonable running time. In the Supplementary File, we present the list of parameters optimized by the PSO algorithm for each aligner. Note that the lower and upper bounds of each parameter were taken from the documentation of each aligner.

In the following experiments, we have employed four alignment benchmarks containing reference alignments: BAliBASE [43], OX-Bench [35], PREFAB [13], and SABmark [44].

BAliBASE 3.0 is one of the most widely-used alignment benchmarks. It defines a total of 218 sets of sequences that are prepared to be aligned by MSA approaches. All the sequences were extracted from the Protein Data Bank [2]. We organized the sets of sequences in six subsets according to their families and similarities: RV11 (38 sets of sequences), RV12 (44), RV20 (41), RV30 (30), RV40 (49), and RV50 (16).

OX-Bench includes a reference database of protein multiple sequence alignments that were generated by a consideration of protein's three-dimensional structure. Out of the 395 reference alignments in OX-Bench, there are 191 alignments that have protein sequences which belong to one or the other of the 43 selected protein folds. In our comparative study, we divide the 395 sets into four groups according to the identity (Id): [0-15%) (6 sets, where $0\% \leq Id < 15\%$), [15-30%) (42 sets, where $15\% \leq Id < 30\%$), [30-50%) (117 sets, where $30\% \leq Id < 50\%$), and [50-100%] (230 sets, where $50\% \leq Id \leq 100\%$).

The Protein REFerence Alignment Benchmark (PREFAB) version 4.0 includes a collection of 1,680 alignments. We divided the 1,680 sets into four groups according to

Table 1: Results in terms of Q-score and TC-score (in %) obtained by the PSO algorithm when optimizing the parameters of five different aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. Each cell of the table includes the value of Q-score/TC-score obtained by each aligner using the default parameters configuration, followed by the improvement obtained by using the best parameter configuration found by PSO algorithm.

	Q-score (in %)					
	RV11	RV12	RV20	RV30	RV40	RV50
Clustal W	50.06 +10.32	86.43 +6.35	85.20 +6.71	72.50 +9.55	78.94 +11.72	74.25 +10.33
DIALIGN-TX	50.47 +6.17	88.21 +2.58	87.81 +2.14	76.14 +5.76	83.40 +5.85	82.15 +3.06
Kalign2	60.53 +10.62	91.21 +2.88	90.08 +2.69	81.26 +5.23	88.33 +4.60	82.01 +5.40
MAFFT	59.47 +13.95	92.41 +3.53	90.04 +3.95	84.47 +2.04	89.88 +3.05	87.64 +0.22
MUSCLE	57.15 +12.67	91.54 +3.30	88.89 +4.17	81.44 +5.36	86.31 +7.28	83.52 +6.14
	TC-score (in %)					
	RV11	RV12	RV20	RV30	RV40	RV50
Clustal W	22.99 +14.14	71.68 +13.15	22.16 +23.43	27.59 +23.94	39.82 +23.40	31.16 +24.60
DIALIGN-TX	26.00 +7.99	73.95 +5.13	30.36 +7.59	38.78 +10.33	44.84 +10.34	46.71 +7.16
Kalign2	36.87 +13.55	79.34 +6.98	36.25 +14.88	47.99 +12.53	50.78 +17.36	43.96 +14.89
MAFFT	37.78 +16.65	83.18 +7.29	38.85 +18.35	51.98 +8.65	54.86 +13.28	56.33 +2.52
MUSCLE	32.06 +13.66	80.90 +6.86	35.30 +16.04	41.19 +18.99	45.32 +26.89	46.39 +14.89

the identity (Id): [0-10%) (102 sets, where $0\% \leq Id < 10\%$), [10-20%) (702 sets, where $10\% \leq Id < 20\%$), [20-40%) (658 sets, where $20\% \leq Id < 40\%$), and [40-100%] (218 sets, where $40\% \leq Id \leq 100\%$).

Finally, SABmark v1.65 consists of 423 sets of sequences. These sets are divided into two families: Superfamily (315 sets) and Twilight (108 sets). The total of 423 sets of sequences was organized in four groups: sup [0-20%) (99 sets from Superfamily, where $0\% \leq Id < 20\%$), sup [20-100%] (216 sets from Superfamily, where $20\% \leq Id \leq 100\%$), twi [0-20%) (78 sets from Twilight, where $0\% \leq Id < 20\%$), and twi [20-100%] (30 sets from Twilight, where $20\% \leq Id \leq 100\%$).

These four benchmarks (BAliBASE, OX-Bench, PREFAB, and SABmark) containing reference alignments were obtained from [14]. In Figure 1, we compare the selected biological characteristics proposed in the previous section for the four alignment benchmarks. As seen, the four benchmarks present different biological characteristics, providing variety to test the accuracy of the aligners as a result.

In the literature, the Q and TC assessment metrics [13] are commonly employed to quantify the agreement between the true alignment and the alignments obtained by the approaches. The former indicates the number of correctly aligned residue pairs divided by the number of residue pairs in the reference alignment. The second metric (TC), is the number of correctly aligned columns divided by the number of columns in the reference alignment. In this work, these metrics were calculated using the *qscore program* [15]. Note that, the TC score is inapplicable to PREFAB as the reference alignments are pairwise [13]; therefore, in our study, for this benchmark, we compare the methods by only using the Q score.

The machine used in the comparative study was a PC with four Intel cores (2.3GHz) with 4GB RAM (Linux Operating System).

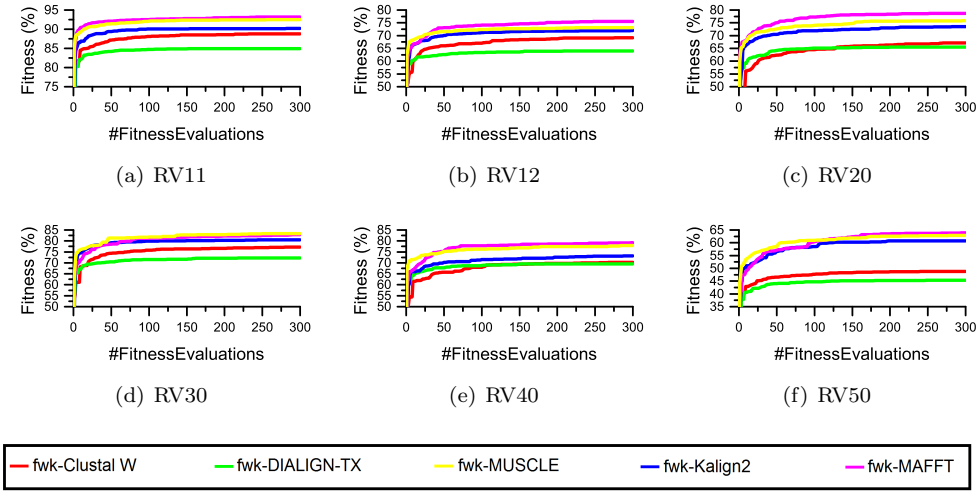


Figure 2: Study of PSO convergence in 300 fitness evaluations. For each subset of BAliBASE v3.0, we present the evolutionary trajectory of fitness obtained by PSO for the five aligners considered in this work: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE.

4.2. Obtaining characteristic-configuration files

As discussed in the previous section, we have five different aligners (Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE). Therefore, we need a *characteristic-configuration* file for each method. To create them, we employed the 218 sets of sequences included in the BAliBASE benchmark. The choice of using BAliBASE builds on the fact that in this benchmark the alignments are based upon protein 3D structure super-positioning and are *manually* refined in order to ensure higher alignment quality than a pure automated method. This property can be seen as one of the main features of BAliBASE, but can also be considered as source of subjectivity due to the *expert* refinement [3]. In addition (as shown in Figure 1), the BAliBASE benchmark presents an homogeneous distribution of its 218 datasets in terms of the identity and number of sequences.

In the PSO algorithm we used a small number of particles; namely a swarm size equals 10. In addition, the maximum number of fitness evaluations was established at 300.

As we may observe in Table 1, the five aligners greatly improve the two metrics (Q and TC). However, we see that the TC-score represents a much greater improvement than the Q-score in all cases. In terms of the average Q-score and TC-score obtained by the aligners in the 218 alignments of BAliBASE, the ranking of the aligners with the default configuration was (Q-score, TC-score): MAFFT, Kalign2, MUSCLE, DIALIGN-TX, and Clustal W. After obtaining the best configuration with the PSO algorithm, the updated ranking is: MAFFT, MUSCLE, Kalign2, Clustal W, and DIALIGN-TX. MAFFT is accordingly still the best performer; yet Clustal W and MUSCLE have scaled in the ranking. It is worth highlighting the great average improvement obtained by Clustal W in both the Q-score (>9%) and TC-score (>20%). Finally, in Figure 2, we

Table 2: Results obtained by 16 aligners when solving a total of 395 datasets included in the OX-Bench benchmark suite. Note that, the 395 datasets are divided into four groups according to identity: [0, 15)%, [15, 30)%, [30, 50)%, and [50, 100]%. In addition, for each group of datasets and each aligner we report the following performance metrics: Q-score (%), TC-score (%), and Runtime (t, in seconds). Note that the three best values of Q-score and TC-score are highlighted in three tones of gray (light, medium, and dark).

	Identity											
	[0, 15)%			[15, 30)%			[30, 50)%			[50, 100]%		
	Q	TC	t	Q	TC	t	Q	TC	t	Q	TC	t
fwk-Clustal W	29.00	11.12	$2.67e^{-2}$	63.53	37.76	$6.08e^{-2}$	87.63	74.62	$1.89e^{-1}$	97.98	94.9	$6.97e^{-2}$
fwk-DIALIGN-TX	13.59	2.39	$1.58e^{-1}$	52.15	23.78	$5.14e^{-1}$	83.64	68.24	$2.89e^{+0}$	97.18	93.29	$7.25e^{-1}$
fwk-Kalign2	27.47	10.98	$1.19e^{-2}$	61.69	36.7	$2.02e^{-2}$	88.45	76.57	$6.33e^{-2}$	97.93	94.88	$1.88e^{-2}$
fwk-MAFFT	26.64	11.84	$2.82e^{-1}$	64.67	40.81	$3.48e^{-1}$	88.48	76.73	$5.72e^{-1}$	98.05	95.17	$3.43e^{-1}$
fwk-MUSCLE	31.39	16.05	$6.43e^{-2}$	64.43	40.72	$1.04e^{-1}$	88.11	76.15	$3.23e^{-1}$	97.96	95.01	$6.11e^{-2}$
Clustal W	27.87	11.12	$1.91e^{-2}$	62.3	35.64	$5.27e^{-2}$	86.02	71.77	$1.73e^{-1}$	97.75	94.41	$5.75e^{-2}$
DIALIGN-TX	12.96	1.25	$1.34e^{-1}$	50.09	22.56	$4.18e^{-1}$	82.05	64.88	$2.73e^{+0}$	96.9	92.69	$6.67e^{-1}$
Kalign2	17.82	3.22	$4.55e^{-3}$	57.05	31.91	$7.28e^{-3}$	85.26	69.95	$8.54e^{-3}$	97.55	93.98	$5.30e^{-3}$
MAFFT	21.58	6.97	$2.64e^{-1}$	59.5	33.75	$3.13e^{-1}$	85.38	70.69	$4.13e^{-1}$	97.41	93.75	$3.13e^{-1}$
MUSCLE	25.1	8.33	$5.47e^{-2}$	61.65	36.38	$9.07e^{-2}$	86.82	73.56	$1.21e^{-1}$	97.62	94.27	$3.83e^{-2}$
FSA	11.19	0	$3.47e^{-1}$	48.05	20.69	$1.70e^{+0}$	82.46	65.37	$3.38e^{+0}$	97.41	93.42	$1.39e^{+0}$
MSAProbs	25.44	14.27	$1.08e^{-1}$	62.27	36.98	$5.21e^{-1}$	87.76	74.95	$2.68e^{+0}$	97.99	95.06	$4.92e^{-1}$
MUMMALS	25.63	7.12	$3.58e^{-1}$	64.58	39.87	$9.71e^{-1}$	87.87	75.17	$9.25e^{-1}$	97.83	94.71	$1.39e^{-1}$
ProbAlign	26.85	12.78	$7.67e^{-2}$	60.7	35.45	$3.11e^{-1}$	87.79	75.12	$1.75e^{+0}$	98.06	95.25	$3.09e^{-1}$
ProbCons	26.29	10.25	$1.27e^{-1}$	61.44	36.27	$5.12e^{-1}$	87.04	73.58	$2.58e^{+0}$	97.83	94.58	$5.00e^{-1}$
T-Coffee	21.07	4.61	$3.57e^{-1}$	61.55	36.94	$1.45e^{+0}$	87.44	74.11	$7.30e^{+0}$	97.92	94.77	$1.37e^{+0}$

show a convergence analysis of PSO for each aligner. As shown, 300 fitness evaluations were sufficient to find the best configuration in all the aligners tested.

4.3. Comparative Study

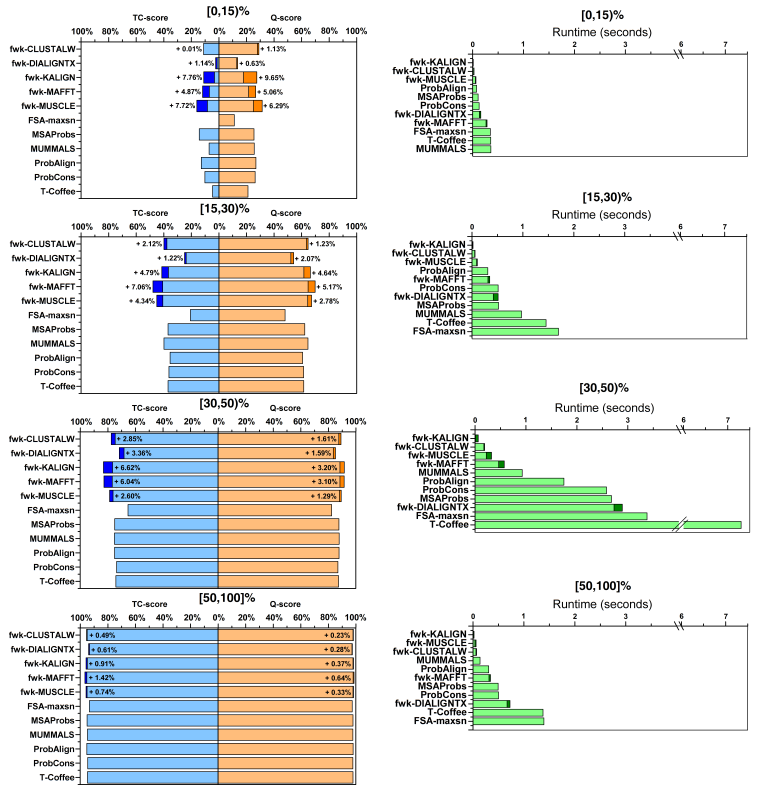
In the previous section, we obtained a characteristic-configuration file for each of the five selected aligners (Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE) by using a PSO algorithm and the BAliBASE benchmark (218 datasets). In the comparative study, to validate the framework, we use the other three alignment benchmarks: OX-Bench (395 datasets), PREFAB (1680 datasets), and SABmark (423 datasets).

The fwk-ClustalW, fwk-DIALIGN-TX, fwk-Kalign2, fwk-MAFFT, and fwk-MUSCLE (characteristic-based version of the aligners) will be compared with the most relevant MSA methods published in the literature: Clustal W [42] (v2.1), DIALIGN-TX [41] (v1.0.2), FSA (with the -maxn option) [4] (v1.15.9), Kalign2 [24] (v2.03), MAFFT [21] (v7.215), MSAProbs [26] (v0.9.7), MUMMALS [34] (version dated on 08/02/2008), MUSCLE [13] (v3.8), ProbAlign [36] (v1.4), ProbCons [9] (v1.12), and T-Coffee [32] (v11.0).

4.3.1. OX-Bench benchmark

We start our comparative study with the OX-Bench benchmark which consists of 395 datasets divided into four groups by identity.

In Table 2, we compare the Q-score, TC-score, and runtime obtained by the default Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE with the results obtained by well-known aligners: FSA, MSAProbs, MUMMALS, ProbAlign, ProbCons, and T-Coffee. As seen, the default parameters versions are not competitive enough, obtaining lower values of the Q-score and TC-score than the well-known aligners. If we compute the average value of the Q-score and TC-score in the 395 datasets of OX-Bench, we obtain



(a) Q-score and TC-score (in %) (b) runtime(in seconds)

Figure 3: Illustrative comparison among the 16 aligners when dealing with the OX-Bench benchmark. In (a), we show the improvements in terms of the Q-score and TC-score (in %) obtained by using the characteristic-based framework with five aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. In (b), we report the runtime overhead introduced by the framework to the aforementioned aligners.

the following ranking: MSAProbs, MUMMALS, ProbAlign, ProbCons, Clustal W, MUSCLE, T-Coffee, MAFFT, Kalign2, DIALIGN-TX, and FSA-maxsn. As shown, Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE obtained a worse performance than MSAProbs, MUMMALS, ProbAlign, and ProbCons on average.

In Figure 3(a), we present a visual representation of the improvement achieved by the characteristic-based versions of the five aligners. The five aligners improve their results in terms of the Q-score and TC-score with a low increment in runtime (see Figure 3(b)). In addition, it is possible to note the great improvements obtained by Clustal W, Kalign2, MAFFT, and MUSCLE.

Finally, the ranking in terms of the average Q-score and TC-score is now: fwk-MUSCLE, fwk-MAFFT, fwk-Clustal W, MSAProbs, fwk-Kalign2, MUMMALS, ProbAlign, ProbCons, T-Coffee, fwk-DIALIGN-TX, and FSA-maxsn. As we may observe, the characteristic-based framework helps MUSCLE, MAFFT, and CLUSTAL W to scale in the OX-Bench ranking, thereby becoming the three best aligners. In addition, Kalign2

Table 3: Results obtained by 16 aligners when solving a total of 1680 datasets included in the PREFAB benchmark suite. Note that, the 1680 datasets are divided into four groups according to identity: [0, 10)%, [10, 20)%, [20, 40)%, and [40, 100]%. In addition, for each group of datasets and each aligner we report the following performance metrics: Q-score (%), and Runtime (t, in seconds). Note that the three best values of Q-score are highlighted in three tones of gray (light, medium, and dark).

	Identity							
	[0, 10)%		[10, 20)%		[20, 40)%		[40, 100]%	
	Q	t	Q	t	Q	t	Q	t
fwk-Clustal W	20.82	$2.99e^{+0}$	47.06	$2.79e^{+0}$	79.38	$4.40e^{+0}$	94.22	$4.20e^{+0}$
fwk-DIALIGN-TX	16.36	$1.71e^{+1}$	46.38	$1.54e^{+1}$	79.89	$1.89e^{+1}$	94.68	$1.76e^{+1}$
fwk-Kalign2	23.99	$1.22e^{+0}$	53.36	$1.04e^{+0}$	83.36	$1.39e^{+0}$	95.09	$1.34e^{+0}$
fwk-MAFFT	26.45	$7.83e^{+0}$	58.78	$5.79e^{+0}$	85.68	$6.00e^{+0}$	95.51	$4.94e^{+0}$
fwk-MUSCLE	25.15	$3.30e^{+0}$	55.67	$2.66e^{+0}$	84.01	$3.94e^{+0}$	94.94	$5.27e^{+0}$
Clustal W	16.40	$2.73e^{+0}$	40.51	$2.52e^{+0}$	75.23	$4.01e^{+0}$	92.33	$3.89e^{+0}$
DIALIGN-TX	13.78	$1.47e^{+1}$	40.94	$1.40e^{+1}$	75.93	$1.73e^{+1}$	93.99	$1.62e^{+1}$
Kalign2	17.56	$1.04e^{-1}$	44.89	$8.85e^{-2}$	77.40	$1.11e^{-1}$	93.78	$1.04e^{-1}$
MAFFT	20.20	$3.02e^{+0}$	51.32	$2.46e^{+0}$	81.73	$2.18e^{+0}$	94.21	$1.56e^{+0}$
MUSCLE	19.31	$2.40e^{+0}$	48.51	$1.86e^{+0}$	80.29	$1.75e^{+0}$	93.21	$1.52e^{+0}$
FSA	2.90	$9.95e^{+1}$	23.17	$8.44e^{+1}$	70.68	$1.19e^{+2}$	94.21	$1.19e^{+2}$
MSAProbs	23.04	$5.14e^{+1}$	56.46	$4.35e^{+1}$	84.91	$5.97e^{+1}$	94.10	$5.76e^{+1}$
MUMMALS	29.32	$7.55e^{+1}$	59.76	$5.06e^{+1}$	85.75	$3.90e^{+1}$	95.81	$1.93e^{+1}$
ProbAlign	19.79	$3.61e^{+1}$	53.71	$3.03e^{+1}$	83.91	$4.20e^{+1}$	94.28	$4.11e^{+1}$
ProbCons	21.01	$6.14e^{+1}$	53.61	$5.21e^{+1}$	83.33	$7.14e^{+1}$	93.57	$6.83e^{+1}$
T-Coffee	20.36	$1.48e^{+2}$	53.50	$1.27e^{+2}$	82.83	$1.76e^{+2}$	93.89	$1.68e^{+2}$

* Q score is in percentage (%) and t refers to the average runtime in seconds

scales from position 9 to position 5 in the ranking. Comparing fwk-MUSCLE (the best characteristic-based version) and MSAProbs (the best well-known aligner), we observe that fwk-MUSCLE not only obtains better Q and TC score, but is also able to solve the 395 datasets of OX-Bench around six times faster than MSAProbs.

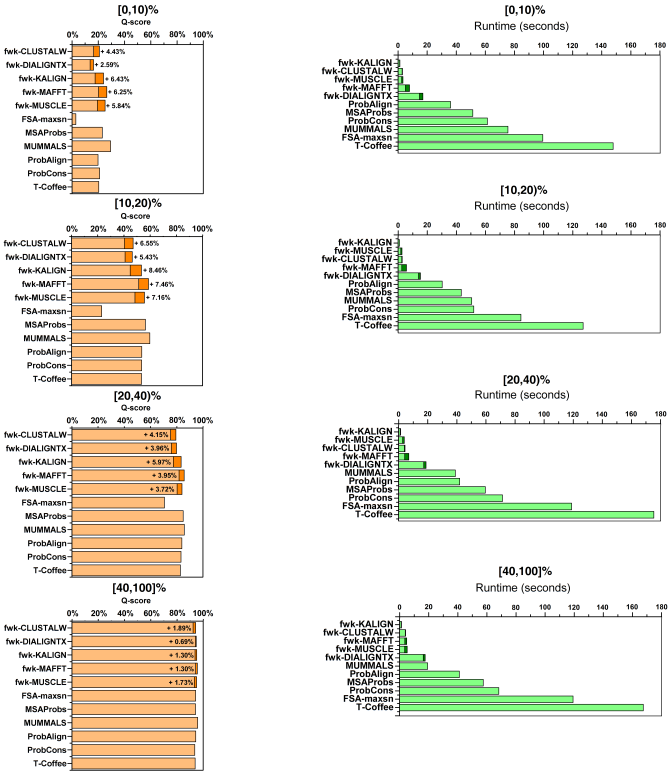
4.3.2. PREFAB benchmark

The second alignment benchmark is PREFAB, which contains a total of 1680 datasets divided into four groups by identity.

Like we did in OX-Bench, we start comparing the Q-score and runtime obtained by the default versions of the five selected aligners (Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE) with the other well-known aligners (see Table 3).

The alignment accuracy of Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE is far from the accuracy obtained by other aligners such as MUMMALS, MSAProbs, or ProbAlign. In this case, the ranking taking into account the average Q-score value in all datasets of PREFAB is as follows: MUMMALS, MSAProbs, ProbAlign, ProbCons, T-Coffee, MAFFT, MUSCLE, Kalign2, DIALIGN-TX, Clustal W, and FSA-maxsn. As occurred in the previous benchmark, the five aligners under study are only better than FSA-maxsn.

If we focus on the characteristic-based versions (fwk-Clustal W, fwk-DIALIGN-TX, fwk-Kalign2, fwk-MAFFT, and fwk-MUSCLE), we see the five aligners tested are able to achieve a great improvement (over 4% in average), as shown in Figure 4(a). The updated ranking in terms of the Q-score is: MUMMALS, fwk-MAFFT, fwk-MUSCLE, MSAProbs, fwk-Kalign2, ProbAlign, ProbCons, T-Coffee, fwk-CLUSTALW, fwk-DIALIGNTX, and FSA-maxsn.



(a) Q-score (in %) (b) runtime(in seconds)

Figure 4: Illustrative comparison among the 16 aligners when dealing with the PREFAB benchmark. In (a), we show the improvements in terms of the Q-score (in %) obtained by using the characteristic-based framework with five aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. In (b), we report the runtime overhead introduced by the framework to the aforementioned aligners.

In this case, we can observe that MAFFT, MUSCLE, and Kalign2 obtain a good position in the final ranking of PREFAB. The best aligner in PREFAB remains MUMMALS; however, in terms of runtime (see Figure 4(b)), if we compare the average running time required by the three best aligners in PREFAB: MUMMALS (46.10 seconds), fwk-MAFFT (6.14 seconds), and fwk-MUSCLE (3.79 seconds); then we can see that fwk-MAFFT and fwk-MUSCLE are approximately 7.5 and 12 times faster than MUMMALS, respectively.

4.3.3. SABmark benchmark

The last alignment benchmark is SABmark (423 datasets), which is divided into two families: Superfamily (315 sets) and Twilight (108 sets); and each family by identity (a total of four groups).

As shown in Table 4, if we compare the Q-score, TC-score, and runtime obtained by the well-known aligners and the default version of Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE, we notice that the default versions obtain a poor level of alignment accuracy compared to the other aligners. The ranking of the aligners taking the

Table 4: Results obtained by 16 aligners when solving a total of 423 datasets included in the SABmark benchmark suite. Note that, the 423 datasets are divided into four groups according to their family and identity: superfamily [0, 25)% and [25, 100)%, twilight [0, 20)% and [20, 100]%. In addition, for each group of datasets and each aligner we report the following performance metrics: Q-score (%), TC-score (%), and Runtime (t, in seconds). Note that the three best values of Q-score and TC-score are highlighted in three tones of gray (light, medium, and dark).

	Identity											
	Superfamily						Twilight					
	[0, 20)%			[20, 100]%			[0, 20)%			[20, 100]%		
	Q	TC	t	Q	TC	t	Q	TC	t	Q	TC	t
fwk-Clustal W	62.05	40.06	$7.17e^{-2}$	62.28	41.83	$7.66e^{-2}$	34.65	16.46	$4.89e^{-2}$	40.41	24.68	$4.09e^{-2}$
fwk-DIALIGN-TX	31.15	7.70	$4.69e^{-1}$	72.29	51.57	$4.37e^{-1}$	21.96	5.65	$3.06e^{-1}$	60.22	37.28	$4.74e^{-1}$
fwk-Kalign2	37.78	14.11	$2.81e^{-2}$	76.70	59.34	$3.05e^{-2}$	28.98	12.06	$1.92e^{-2}$	66.51	49.71	$2.81e^{-2}$
fwk-MAFFT	40.58	14.66	$4.72e^{-1}$	78.92	62.31	$4.90e^{-1}$	33.34	14.18	$3.94e^{-1}$	69.30	51.00	$5.69e^{-1}$
fwk-MUSCLE	40.18	14.24	$2.03e^{-1}$	77.13	59.16	$1.50e^{-1}$	31.55	11.80	$1.52e^{-1}$	66.54	45.88	$1.93e^{-1}$
Clustal W	58.70	35.49	$5.90e^{-2}$	59.12	37.86	$6.19e^{-2}$	29.98	12.81	$4.06e^{-2}$	35.28	21.03	$3.33e^{-2}$
DIALIGN-TX	28.50	6.17	$3.40e^{-1}$	70.23	48.13	$3.75e^{-1}$	19.66	4.42	$2.27e^{-1}$	56.35	32.78	$3.46e^{-1}$
Kalign2	30.26	10.44	$9.12e^{-3}$	71.34	51.20	$8.66e^{-3}$	23.55	9.23	$6.75e^{-3}$	60.25	41.33	$8.00e^{-3}$
MAFFT	33.53	10.68	$4.00e^{-1}$	75.08	55.81	$4.10e^{-1}$	24.98	8.44	$3.57e^{-1}$	66.05	44.54	$5.59e^{-1}$
MUSCLE	34.52	9.48	$1.96e^{-1}$	73.57	52.72	$1.40e^{-1}$	24.86	8.07	$1.47e^{-1}$	60.27	40.07	$1.86e^{-1}$
FSA	29.88	6.04	$2.26e^{+0}$	74.86	54.46	$2.82e^{+0}$	23.04	6.69	$1.14e^{+0}$	62.99	38.50	$2.39e^{+0}$
MSAProbs	37.68	11.32	$5.49e^{-1}$	79.28	61.82	$6.66e^{-1}$	31.71	11.53	$2.83e^{-1}$	71.80	52.16	$5.71e^{-1}$
MUMMALS	41.01	14.77	$2.50e^{+0}$	80.49	64.04	$3.26e^{+0}$	34.33	13.50	$1.10e^{+0}$	72.09	52.99	$2.95e^{+0}$
ProbAlign	36.55	10.00	$3.95e^{-1}$	78.62	59.74	$4.74e^{-1}$	31.72	11.18	$2.13e^{-1}$	70.26	52.46	$4.20e^{-1}$
ProbCons	36.69	10.53	$7.11e^{-1}$	78.74	60.69	$8.56e^{-1}$	31.19	11.07	$3.77e^{-1}$	72.10	51.67	$7.49e^{-1}$
T-Coffee	36.68	11.44	$1.95e^{+0}$	79.05	61.49	$2.27e^{+0}$	30.43	11.99	$1.11e^{+0}$	73.89	56.42	$2.00e^{+0}$

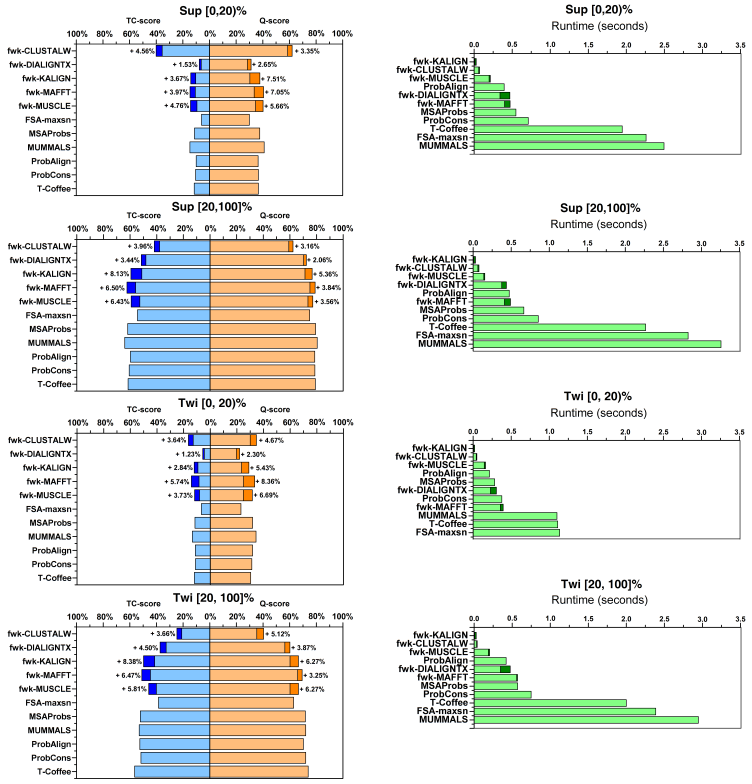
average value of the Q-score and TC-score into account is as follows: MUMMALS, T-Coffee, MSAProbs, ProbCons, ProbAlign, MAFFT, MUSCLE, Kalign2, FSA-maxsn, Clustal W, and DIALIGN-TX. As in our previous experiments, the five aligners under study obtained the worst alignment performance in SABmark, highlighting the cases of Clustal W, and DIALIGN-TX that are ranked in the last positions.

Applying our characteristic-based framework to Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE reveals a great accuracy improvement for fwk-Kalign2, fwk-MAFFT, and fwk-MUSCLE (over 5% in terms of the Q-score and TC-score). fwk-Clustal W and fwk-DIALIGN-TX also undergo a moderate improvement. In Figure 5(a), we present a visual comparison of the improvements obtained by each of the five characteristic-based aligners. We can highlight the particularly good performance of fwk-Clustal W in those datasets with a low percentage of identity (<20%).

Finally, the updated ranking is: MUMMALS, fwk-MAFFT, T-Coffee, MSAProbs, ProbCons, ProbAlign, fwk-MUSCLE, fwk-Kalign2, fwk-Clustal W, and FSA-maxsn. Analyzing the ranking, we observe that the three best aligners (in Q-score and TC-score terms) are now: MUMMALS, fwk-MAFFT, and T-Coffee. However, if we focus on Figure 5(b), we notice that fwk-MAFFT is around 4 and 5 times faster than T-Coffee and MUMMALS, respectively.

4.4. Discussion

In the previous section, we studied the benefits of using our characteristic-based framework with five different aligners (Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE) in three alignment benchmarks: OX-Bench (395 datasets), PREFAB (1,680 datasets), and SABMark (423 datasets).



(a) Q-score and TC-score (in %) (b) runtime(in seconds)

Figure 5: Illustrative comparison among the 16 aligners when dealing with the SABmark benchmark. In (a), we show the improvements in terms of the Q-score and TC-score (in %) obtained by using the characteristic-based framework with five aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. In (b), we report the runtime overhead introduced by the framework to the aforementioned aligners.

We have seen that, the characteristic-based framework provides reasonable accuracy and conservation improvements to the aligners. For each of the three alignment benchmarks and for each of the five tested aligners, in Figure 6 we present the percentage of datasets in which the characteristic-based framework obtains significant accuracy/conservation improvements over the default configuration. In OX-Bench benchmark, we can see that on average, fwk-MAFFT is the aligner that obtains the highest percentage of datasets improved (65.5%); on the contrary, fwk-DIALIGN-TX is only able to improve 37.05% of the datasets contained in OX-Bench. If we focus on the second benchmark (PREFAB), in this case fwk-Clustal W improves the default configuration in 806 datasets of PREFAB (around 48% of datasets). Finally, in the SABMark benchmark, we observe that fwk-Kalign2, fwk-MAFFT, and fwk-MUSCLE are the aligners with the best percentage of improvement, above 60%.

In Table 5, we summarize the behaviour of the five aligners in the three alignment benchmarks. As shown, the five aligners improve their results in the three benchmarks

Table 5: Summary of the results obtained by the five aligners under study (Clustal W, Kalign2, DIALIGN-TX, MAFFT, and MUSCLE) and their related characteristic-based versions when solving the three selected benchmark suites (OX-Bench, PREFAB, and SABMark). For each benchmark and approach, we report the following performance metrics: Q-score (%), TC-score (%), and Runtime (t, in seconds).

	OX-Bench			PREFAB		SABMark		
	Q	TC	t	Q	t	Q	TC	t
Clustal W	68.48	53.23	7.56E-02	56.12	3.29E+00	45.77	26.80	4.87E-02
fwk-Clustal W	69.53	54.70	8.64E-02	60.37	3.60E+00	49.85	30.76	5.95E-02
DIALIGN-TX	60.50	45.34	9.87E-01	56.16	1.56E+01	43.69	22.87	3.22E-01
fwk-DIALIGN-TX	61.64	46.92	1.07E+00	59.33	1.72E+01	46.41	25.55	4.21E-01
Kalign2	64.42	49.76	6.42E-03	58.41	1.02E-01	46.35	28.05	8.13E-03
fwk-Kalign2	68.88	54.78	2.86E-02	63.95	1.25E+00	52.49	33.80	2.65E-02
MAFFT	65.97	51.29	3.26E-01	61.87	2.30E+00	49.91	29.87	4.31E-01
fwk-MAFFT	69.46	56.13	3.86E-01	66.61	6.14E+00	55.54	35.54	4.81E-01
MUSCLE	67.80	53.13	7.62E-02	60.33	1.88E+00	48.30	27.58	1.67E-01
fwk-MUSCLE	70.47	56.98	1.38E-01	64.94	3.79E+00	53.85	32.77	1.75E-01

with a small increment in running time. On the one hand, we observe that in terms of alignment accuracy and conservation, the ranking, in descending order, is: fwk-MAFFT, fwk-MUSCLE, fwk-Kalign2, fwk-Clustal W, and fwk-DIALIGN-TX. However, if we focus on running time, the fastest algorithms are: fwk-Kalign2, fwk-Clustal W, fwk-MUSCLE, fwk-MAFFT, and fwk-DIALIGN-TX. From these rankings, we can say that the aligner with the best trade-off between acuracy/conservation and runtime is fwk-MUSCLE, while, the worst aligner is clearly DIALIGN-TX.

All in all, we can conclude that the proposed characteristic-based framework is a good option for boosting the alignment accuracy of well-known aligners without excessively penalizing their runtime.

5. Conclusions and Future work

A characteristic-based framework for improving the alignment of multiple sequence aligners was proposed and tested in this paper. This framework studies the input set' s biological characteristics and then applies the best parameter configuration found depending on those characteristics. In this way, better alignments (with better accuracy and conservation) are obtained. The framework uses a pre-computed file to take the best configuration for a dataset with similar biological characteristics. In order to create this file, we use a Particle Swarm Optimization (PSO) algorithm, that is, an algorithm based on swarm intelligence, for finding the best parameters configuration of a given aligner.

After explaining the characteristic-based framework, we presented a comparative study in which we applied the framework to five well-known aligners: Clustal W, DIALIGN-TX, Kalign2, MAFFT, and MUSCLE. In order to obtain a configuration-file for each aligner, we employed a Particle Swarm Optimization algorithm and the BAliBASE v3.0 alignment benchmark. Then, we studied the improvements provided by the framework by using other three benchmarks: OX-Bench, PREFAB v4.0, and SABmark v1.65. In the comparative study, we analyzed the results obtained by the default version and by the characteristic-based version of the five aligners, while comparing their results with other aligners published in the literature, such as FSA (with the -maxn option, v1.15.9),

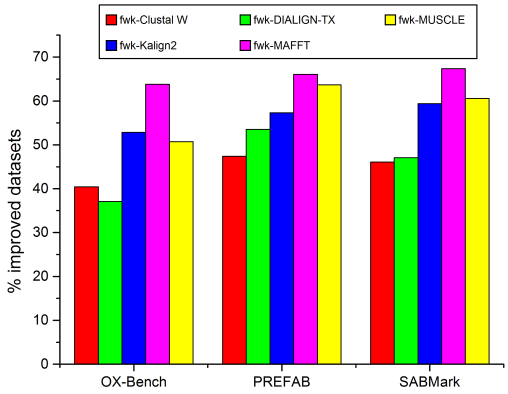


Figure 6: Percentage of datasets in OX-Bench, PREFAB, and SABMark in which the aligners under study (Clustal W, Kalign2, DIALIGN-TX, MAFFT, and MUSCLE) obtain better values of Q-score or TC-score when using the parameters configuration proposed by the characteristic-based framework instead of using the default parameters configuration.

MSAProbs (v0.9.7), MUMMALS (version dated on 08/02/2008), ProbAlign (v1.4), ProbCons (v1.12), and T-Coffee (v11.0). We can conclude that the characteristic-based framework provides significant accuracy improvements to the tested aligners with a reasonable rise in running time.

As demonstrated in a series of recent publications (see, e.g., [19, 20]) on developing new analysis methods, user-friendly, and publicly accessible web-servers will significantly enhance their impacts [8], we shall make efforts in our future work to provide a web-server for our characteristic-based framework. In addition, we intend to tackle the optimization of the parameters by using multiobjective optimization (e.g. [38]), optimizing the Q-score, TC-score, and running time simultaneously. Another important line of future work is to incorporate 3D structure characteristics into the framework so as to add specific knowledge of the problem. Since amino acids have many other physical and chemical properties, a challenging line of future work will be to study the influence of using new amino acids properties to classify the input sequences within the proposed characteristic-based framework; for example, by using a partition map [45] to increase the evolutionary significance at the amino acid sequence level.

Acknowledgments

This work was partially funded by the AEI (State Research Agency, Spain) and the ERDF (European Regional Development Fund, EU), under contract TIN2016-76259-P (PROTEIN project). Álvaro Rubio-Largo is supported by post-doctoral fellowship SFRH/BPD/100872/2014 granted by Fundação para a Ciência e a Tecnologia (FCT), Portugal.

References

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

[2] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, P.E. Bourne, The protein data bank, Nucleic Acids Research 28 (2000) 235–242.
[3] G. Blackshields, I.M. Wallace, M. Larkin, D.G. Higgins, Analysis and comparison of benchmarks for multiple sequence alignment, In silico biology, Journal of Biological Systems Modeling and Simulation 6 (2006) 321 – 339.
[4] R.K. Bradley, A. Roberts, M. Smoot, S. Juvekar, J. Do, C. Dewey, I. Holmes, L. Pachter, Fast statistical alignment, PLoS Computational Biology 5 (2009) e1000392.
[5] W. Chen, H. Lin, K.C. Chou, Pseudo nucleotide composition or PseKNC: an effective formulation for analyzing genomic sequences, Mol. BioSyst. 11 (2015) 2620–2634.
[6] K.C. Chou, Prediction of protein cellular attributes using pseudo-amino acid composition, Proteins: Structure, Function, and Genetics 43 (2001) 246–255.
[7] K.C. Chou, Some remarks on protein attribute prediction and pseudo amino acid composition, Journal of Theoretical Biology 273 (2011) 236–247.
[8] K.C. Chou, Impacts of bioinformatics to medicinal chemistry, Medicinal Chemistry 11 (2015) 218–234.
[9] C.B. Do, M.S. Mahabhashyam, M. Brudno, S. Batzoglou, ProbCons: Probabilistic consistency-based multiple sequence alignment, Genome Research 15 (2005) 330–340.
[10] H. Dogan, H.H. Otu, Objective Functions, Multiple Sequence Alignment Methods, Methods in Molecular Biology (David J. Russell Ed.) 1079 (2014) 45–58.
[11] R. Doolittle, Similar amino acid sequences: chance or common ancestry?, Science 214 (1981) 149–159.
[12] J. Ebert, D. Brutlag, Development and validation of a consistency based multiple structure alignment algorithm, Bioinformatics 22 (2006) 1080–1087.
[13] R.C. Edgar, MUSCLE: multiple sequence alignment with high accuracy and high throughput, Nucleic Acids Research 32 (2004) 1792–1797.
[14] R.C. Edgar, BENCH: A collection of protein sequence alignment benchmarks including BALIBASE v3, PREFAB v4, OXBENCH, and SABRE, *http* : *//www.drive*5*.com/bench*, 2015.
[15] R.C. Edgar, QSCORE: A quality scoring program that compares two multiple sequence alignments, *http* : *//www.drive*5*.com/qscore*, 2015.
[16] D. Feng, R. Doolittle, Progressive sequence alignment as a prerequisite to correct phylogenetic trees, J. Mol. Evolut. 25 (1987) 351–360.
[17] S.H. Guo, E.Z. Deng, L.Q. Xu, H. Ding, H. Lin, W. Chen, K.C. Chou, iNuc-PseKNC: a sequence-based predictor for predicting nucleosome positioning in genomes with pseudo k-tuple nucleotide composition, Bioinformatics 30 (2014) 1522–1529.
[18] P. Hogeweg, B. Hesper, The alignment of sets of sequences and the construction of phyletic trees: an integrated method, Journal of Molecular Biology 20 (1984) 175–186.
[19] J. Jia, Z. Liu, X. Xiao, B. Liu, K.C. Chou, iCar-PseCp: identify carbonylation sites in proteins by monte carlo sampling and incorporating sequence coupled effects into general PseAAC, Oncotarget 7 (2016) 34551–34570.
[20] J. Jia, L. Zhang, Z. Liu, X. Xiao, K.C. Chou, pSumo-CD: predicting sumoylation sites in proteins with covariance discriminant algorithm by incorporating sequence-coupled effects into general PseAAC, Bioinformatics 32 (2016) 3133–3141.
[21] K. Katoh, K. Misawa, K. Kuma, T. Miyata, MAFFT: a novel method for rapid multiple sequence alignment based on fast fourier transform, Nucleic Acids Research 30 (2002) 3059–3066.
[22] J. Kennedy, R. Eberhart, Particle swarm optimization, IEEE International Conference on Neural Networks 4 (1995) 1942–1948.
[23] M. Kimura, A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences, Journal of Molecular Evolution 16 (1980) 111–120.
[24] T. Lassmann, O. Frings, E.L.L. Sonnhammer, Kalign2: high-performance multiple alignment of protein and nucleotide sequences allowing external features, Nucleic Acids Research 37 (2009) 858–865.
[25] B. Liu, H. Wu, K.C. Chou, Pse-in-one 2.0: An improved package of web servers for generating various modes of pseudo components of DNA, RNA, and protein sequences, Natural Science 09 (2017) 67–91.
[26] Y. Liu, B. Schmidt, D.L. Maskell, MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities, Bioinformatics 26 (2010) 1958–1964.
[27] H. Lodish, Molecular Cell Biology, W. H. Freeman, 2008.
[28] A. Loytynoja, 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

(2005) 10557–10562.
[29] F. Naznin, R. Sarker, D. Essam, Vertical decomposition with genetic algorithm for multiple sequence alignment, BMC Bioinformatics 12 (2011) 353.
[30] F. Naznin, R. Sarker, D. Essam, Progressive Alignment Method Using Genetic Algorithm for Multiple Sequence Alignment, IEEE Transactions on Evolutionary Computation 16 (2012) 615–631.
[31] C. Notredame, Recent progresses in multiple sequence alignment: a survey, Pharmacogenomics 3 (2002) 131–144.
[32] C. Notredame, D.G. Higgins, J. Heringa, T-Coffee: a novel method for fast and accurate multiple sequence alignment, Journal of Molecular Biology 302 (2000) 205 – 217.
[33] F.M. Ortuño, O. Valenzuela, F. Rojas, H. Pomares, J.P. Florido, J.M. Urquiza, I. Rojas, Optimizing multiple sequence alignments using a genetic algorithm based on three objectives: structural information, non-gaps percentage and totally conserved columns, Bioinformatics 29 (2013) 2112–2121.
[34] J. Pei, N.V. Grishin, MUMMALS: multiple sequence alignment improved by using hidden markov models with local structural information, Nucleic Acids Research 34 (2006) 4364–4374.
[35] G. Raghava, S. Searle, P. Audley, J. Barber, G. Barton, Oxbench: A benchmark for evaluation of protein multiple sequence alignment accuracy, BMC Bioinformatics 4 (2003) 1–23.
[36] U. Roshan, D.R. Livesay, Probalign: multiple sequence alignment using partition function posterior probabilities, Bioinformatics 22 (2006) 2715–2721.
[37] A. Rubio-Largo, L. Vanneschi, M. Castelli, M.A. Vega-Rodríguez, A characteristic-based framework for multiple sequence aligners, IEEE Transactions on Cybernetics 48 (2018) 41–51.
[38] A. Rubio-Largo, Q. Zhang, M.A. Vega-Rodríguez, A multiobjective evolutionary algorithm based on decomposition with normal boundary intersection for traffic grooming in optical networks, Information Sciences 486 (2014) 110–118.
[39] A. Rubio-Largo, M.A. Vega-Rodríguez, D.L. González-Álvarez, Hybrid multiobjective artificial bee colony for multiple sequence alignment, Applied Soft Computing 41 (2016) 157–168.
[40] A. Rubio-Largo, M.A. Vega-Rodríguez, D.L. González-Álvarez, A hybrid multiobjective memetic metaheuristic for multiple sequence alignment, IEEE Transactions on Evolutionary Computation 20 (2016) 499–514.
[41] A.R. Subramanian, M. Kaufmann, B. Morgenstern, DIALIGN-TX: greedy and progressive approaches for segment-based multiple sequence alignment, Algorithms for Molecular Biology 3:6 (2008).
[42] J.D. Thompson, D.G. Higgins, 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 (1994) 4673–4680.
[43] J.D. Thompson, P. Koehl, O. Poch, BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark, Proteins 61 (2005) 127–136.
[44] I. Van Walle, I. Lasters, L. Wyns, SABmark – A benchmark for sequence alignment that covers the entire known fold space, Bioinformatics 21 (2005) 1267–1268.
[45] C. Yu, S.Y. Cheng, R.L. He, S.S.T. Yau, Protein map: An alignment-free sequence comparison method based on various properties of amino acids, Gene 486 (2011) 110–118.