A Characteristic-Based Framework for Multiple Sequence Aligners

Álvaro Rubio-Largo, Leonardo Vanneschi, Mauro Castelli

NOVA Information Management School, Universidade Nova de Lisboa, Lisbon, Portugal

Miguel A. Vega-Rodríguez

Department of Computer and Communications Technologies, University of Extremadura, Caceres, Spain

This is the final, accepted version of the article published by *IEEE*

Rubio-Largo, Á., Vanneschi, L., Castelli, M., & Vega-Rodriguez, M. A. (2018). A Characteristic-Based Framework for Multiple Sequence Aligners. IEEE Transactions on Cybernetics, 48(1), 41-51, (advanced online publication on 2 october 2016). DOI: 10.1109/TCYB.2016.2621129

A Characteristic-Based Framework for Multiple Sequence Aligners

Álvaro Rubio-Largo, Leonardo Vanneschi, Mauro Castelli, and Miguel A. Vega-Rodríguez

***Abstract*—The multiple sequence alignment is a well-known bioinformatics problem that consists in the alignment of three or more biological sequences (protein or nucleic acid). In the literature, a number of tools have been proposed for dealing with this biological sequence alignment problem, such as progressive methods, consistency-based methods, or iterative methods; among others. These aligners often use a default parameter configuration for all the input sequences to align. However, the default configuration is not always the best choice, the alignment accuracy of the tool may be highly boosted if specific parameter configurations are used, depending on the biological characteristics of the input sequences. In this paper, we propose a characteristic-based framework for multiple sequence aligners. The idea of the framework is, given an input set of unaligned sequences, extract its characteristics and run the aligner with the best parameter configuration found for another set of unaligned sequences with similar characteristics. In order to test the framework, we have used the well-known multiple sequence comparison by log-expectation (MUSCLE) v3.8 aligner with different benchmarks, such as benchmark alignments database v3.0, protein reference alignment benchmark v4.0, and sequence alignment benchmark v1.65. The results shown that the alignment accuracy and conservation of MUSCLE might be greatly improved with the proposed framework, specially in those scenarios with a low percentage of identity. The characteristic-based framework for multiple sequence aligners is freely available for downloading at http://arco.unex.es/arl/fwk-msa/cbf-msa.zip**

***Index Terms*—Characteristics-based, multiple sequence alignment (MSA), particle swarm optimization (PSO).**

I. Introduction

MULTIPLE sequence alignment (MSA) [1] is the simultaneous alignment among three or more biological sequences (nucleotides or amino acids). The main goal of MSA is to reflect biological relationships among different sequences, that is, MSA is an important first step to infer phylogenetics relationships [2], [3]. In addition, MSA is commonly used to provide biological facts about proteins—a well-conserved alignment implies further biological signifi- cance [4]. Therefore, an accurate MSA is highly valuable in the formulation and test hypotheses about protein 3-D structure and function, because it allows us to determine those regions of a gene that are susceptible to mutation.

Besides the classical exact methods, such as dynamic programming, a number of approaches for solving the MSA problem has been proposed in the literature. We can distinguish three main groups: 1) progressive-based methods; 2) consistency-based methods; and 3) iterative refinement methods.

The *progressive-based* methods were the first practical strategies applied to this problem [5]. They are a fundamental pillar in recent MSA approaches. *Progressive-based* 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. Among the most important *progressive-based* tools are: Clustal W [6], PRANK [7], fast statistical alignment (FSA) [8], Kalign [9], and DIALIGN-TX [10].

The *consistency-based* methods 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. The main *consistency-based* methods published are: tree-based consistency objective function for alignment evaluation (T-Coffee) [11], probabilistic consistency-based MSA (ProbCons) [12], and MSAProbs [13].

The idea behind the *iterative refinement* methods is to fix any possible erroneous gap inserted at an early stage of a progressive alignment. These methods start by constructing an initial alignment by using a progressive method; then, the obtained guide-tree is divided into two subtrees which are realigned in order to improve the accuracy of the alignment. This step is repeated until no further improvements are found or until a predefined number of iterations is reached. In the literature, we find the following iterative refinement methods: multiple sequence comparison by log-expectation (MUSCLE) [14], multiple alignment using fast Fourier transform (MAFFT) [15], ProbCons [12] (it allows the option of a final iterative refinement), and MUMMALS [16].

Manuscript received November 24, 2015; revised April 22, 2016 and June 20, 2016; accepted October 21, 2016. Date of publication November 2, 2016; date of current version November 15, 2017. This work was supported by the Spanish Ministry of Economy and Competitiveness and the European Regional Development Fund under Contract TIN2016-76259-P (PROTEIN Project). The work of Á. Rubio-Largo was supported by the Post-Doctoral Fellowship by Fundação para a Ciência e a Tecnologia, Portugal, under Grant SFRH/BPD/100872/2014. This paper was recom- mended by Associate Editor F. Herrera.

Á. Rubio-Largo, L. Vanneschi, and M. Castelli are with the NOVA Information Management School, Universidade Nova de Lisboa, 1070-312 Lisbon, Portugal (e-mail: arl@unex.es; lvanneschi@novaims.unl.pt; mcastelli@novaims.unl.pt).

M. A. Vega-Rodríguez is with the Department of Computer and Communications Technologies, University of Extremadura, 10003 Caceres, Spain (e-mail: mavega@unex.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2016.2621129

All these methods proposed a default parameter configuration for reasonably aligning any input set of unaligned sequences with a certain level of accuracy and conservation. However, the default configuration is far from being the best choice, the alignment accuracy and conservation of the method may be highly boosted if an specific parameter configuration is used, taking into account the biological characteristics of the input sequences. In this paper, we propose a characteristic-based framework for improving the performance of any multiple sequence aligner. The input of the framework are two files: 1) *set of unaligned sequences* (dataset) and 2) *characteristics-configuration file*. The *characteristics-configuration file* depends on the aligner, and contains the best parameter configuration of the aligner for a number of datasets with different characteristics. Therefore, our proposed framework runs the aligner with the best parameter configuration found for a dataset with similar characteristics, boosting the alignment accuracy and conservation as a result.

All in all, the major contributions of this paper include the following.

1) A characteristic-based framework for improving the accuracy and conservation of any MSA method.
2) A description of characteristics for describing the properties of biological sets of unaligned sequences.
3) A case-study that proves the effectiveness of the framework by using the well-known MUSCLE.

The rest of this paper is organized as follows. In Section II, we present the MSA problem. A detailed description of the characteristic-based framework is presented in Section III. Section IV is devoted to present a case-study for proving the power of the proposed framework and a comparison with other approaches published in the literature. Finally, the conclusions and future works are presented in Section V.

II. Multiple Sequence Alignment

MSA is an alignment of more than two sequences and is considered an NP-complete optimization problem [17]. The MSA problem can be defined as follows: given a set of sequences $S : \{s_1, s_2, \ldots, s_N\}$ of lengths $|s_1|, |s_2|, \ldots, |s_N|$ defined over an alphabet Σ (e.g., the nucleotides or the amino-acids alphabets), an MSA of S is defined as $S^* : \{s^*_1, s^*_2, \ldots, s^*_N\}$ where $|s_1| = |s_2| = \ldots = |s_N|$ and S^* is defined over the alphabet $\Sigma \cup \{-\}$ with $\{-\}$ denoting an additional gap symbol.

In this way, a multiple alignment is obtained by adding gaps to the sequences of S so that their lengths become the same. By considering a matrix representation, rows correspond to sequences, while columns represent aligned symbols. Each column of an alignment must contain at least one symbol of Σ, (i.e., a column with all gaps is not allowed). According to [17], the complexity of finding an optimal alignment is $O(N2^N L^N)$, where N is the number of sequences and L is the $\max(|s_1|, |s_2|, \ldots, |s_N|)$.

In the following, we present an example of MSA.

1) *Unaligned Sequences (Input):*
   s_1: `GDNISDDEDEV (11).`
   s_2: `KQLTQDDDTDEVEIAVD (17).`
   s_3: `ACRKNDDGDTVVVVEKDHFMDDFF (24).`
2) *Aligned Sequences (Output):*
   s^*_1: `GDNI-SDDEDEV (25).`
   s^*_2: `KQLTQDDDTDEV-EIAVD (25).`
   s^*_3: `ACRK-NDDGDTVVVVEKDHFMDDFF (25).`

To assess the quality of a test alignment with respect to a reference alignment, we take into account two well-known scoring methods [14].

1) *Q-Score (Also Known as Sum-of-Pairs Score):* Let us consider a test alignment of N sequences consisting of M columns. The ith column in the alignment is denoted by $A_{i1}, A_{i2}, \ldots, A_{ik}$. For each pair of residues A_{ij} and A_{ik}, we define p_{ijk} such that $p_{ijk} = 1$ if the residues A_{ij} and A_{ik} are aligned with each other in the reference alignment; otherwise, $p_{ijk} = 0$. The score q_i for the ith column is defined as

$$q_i = \sum_{j=1}^{N} \sum_{k=1, j\neq=1}^{N} p_{ijk}. \quad (1)$$

   The Q-score for the alignment is equal to

$$Q = \frac{\sum_{i=1}^{M} q_i}{\sum_{i=1}^{Mr} qr_i} \quad (2)$$

   where Mr is the number of columns in the reference alignment and qr_i is the score q_i for the *ith* column in the reference alignment.
2) *Total Column Score (TC-Score):* Let us define a C_i score for the ith column in the alignment: $C_i = 1$ if all the residues are aligned with the reference alignment; otherwise, $C_i = 0$. The TC-score for the test alignment is then calculated as follows:

$$TC = \frac{\sum_{i=1}^{M} C_i}{M}. \quad (3)$$

III. Methods

In this section, we describe the characteristic-based framework, detailing all the required implementation aspects.

In the first place, we describe the main characteristics that will be extracted from a set of unaligned sequences. A total of ten characteristics have been selected. We divide them in three different groups.

1) Characteristics A:
   A1. Number of unaligned sequences.
   A2. Average length of the unaligned sequences.
   A3. Standard deviation of the length.
2) Characteristics B:
   B1. Average Kimura distance (evolutionary distance) between each pair of unaligned sequences.
   B2. Standard deviation of the Kimura distance.
3) Characteristics C:
   C1. Percentage of amino-acids with electrically charged side chains (positive): `R`, `H`, and `K`.
   C2. Percentage of amino-acids with electrically charged side chains (negative): `D` and `E`.
   C3. Percentage of amino-acids with polar uncharged side chains: `S`, `T`, `N`, and `Q`.

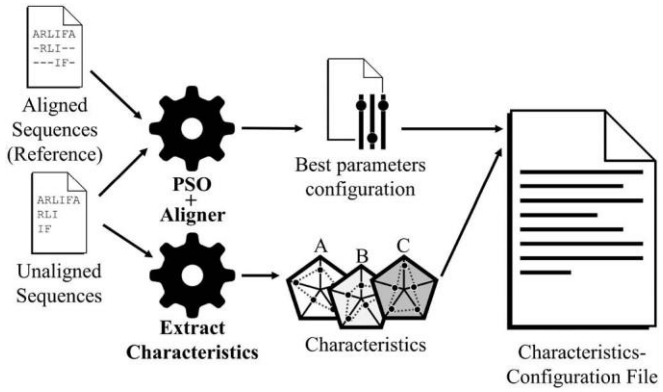

Fig. 1. Generation of a new line (characteristic and its corresponding parameter configuration) to be stored in the *characteristic-configuration* file by using a PSO algorithm.

C4. Percentage of special amino-acids cases: `C`, `U`, `G`, and `P`.

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

The features within the group A refers to the number and length of the unaligned sequences. Within the group B, we find the features regarding the evolutionary distance between each pair of unaligned sequences (average and standard deviation). Finally, in the group C we find the side-chain features (in %): weak acid (positive), weak base (negative), polar, special, and hydrophobic.

As we mentioned before, the characteristic-based framework receives an input file that we denote as *characteristic-configuration* file. This file stores several independent lines, where each line consists of the ten aforementioned biological characteristics and the best-parameters-configuration of the input aligner. The line structure is

`A1-A3;B1-B2;C1-C5;<configuration>.`

In this paper, to obtain the best parameter configuration of a given set of unaligned sequences, we used the particle swarm optimization algorithm (PSO, [18]) because it is a well-known evolutionary algorithm based on swarm intelligence that has demonstrated over the years a very good behavior on discrete [19] and continuous [20], [21] optimization problems. However, any approach may be used for configuring the aligners, such as the statistical method *i*-race [22], some other specific metaheuristics like ParamILS [23], or other evolutionary approaches based on information learning like the artificial bee colony algorithm based on information learning (ILABC) [24] or genetic learning PSO (GL-PSO) [25].

The input parameters of the PSO are: set of unaligned sequences, reference aligned set of sequences (true alignment), aligner binary program, list of parameters to optimize, and upper and lower bounds of the parameters. In the PSO, to quantify the agreement between the true alignment and the alignments obtained by the aligner, two measures of accuracy (Q and TC [14]) are employed to maximize the objective function

$$\text{ObjectiveFunction} = (0.5 * Q) + (0.5 * \text{TC}). \quad (4)$$

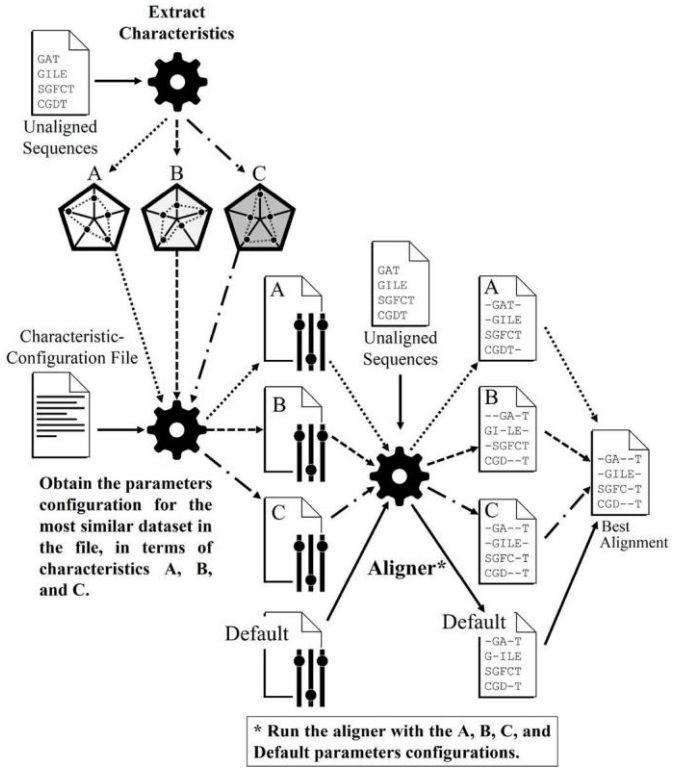

Fig. 2. Illustrative procedure of the characteristic-based framework for multiple sequence aligners.

Finally, the output of the PSO algorithm will be the parameter configuration that produces the closest alignment to the given true alignment. In Fig. 1, we illustrate the procedure to generate a new line (characteristic and its corresponding parameter configuration) to be stored in the *characteristic-configuration* file.

In Fig. 2, we present the illustrative procedure of the characteristic-based framework. In the following, we describe the steps of the framework.

Step 1: Extract the characteristics of the input set of unaligned sequences, that is to say, the three groups: A, B, and C.

Step 2: Normalize the characteristics A, B, and C of the input set and also the characteristics A, B, and C included in the *characteristic-configuration* file. Then, compute the Euclidean distance between the characteristics A of the input set of sequences and every configuration stored in the *characteristic-configuration* file, with the aim of obtaining the closest parameter configuration in terms of characteristics A, we refer to it as P_A. Repeat the process with the other two groups of characteristics (B and C), in order to obtain two more parameter configurations (P_B and P_C).

Step 3: Run the aligner with P_A, P_B, P_C and also with the default parameter configuration.

Step 4: Evaluate the four alignments, and return the alignment with the highest accuracy and conservation.

Running the aligner with the default parameter options ensures a certain level of alignment accuracy, at least as good

TABLE I
PARAMETERS OF MUSCLE

Parameter	Description	Values
−maxiters	Maximum number of iterations (default=16).	[1-20]
−maxtrees	Maximum number of new trees to build in iteration 2 (default=1).	[1-5]
profile	Profile score (default −le).	{−le, −sp, −sv}
diagonal	Diagonal optimization (default none).	{none, −diags1, −diags2, −diags}
anchor	Anchor Optimization (default −noanchor).	{−noanchor, −anchors}
−cluster1	Cluster method in iteration 1 (default upgmb).	{upgma, upgmb, neighborjoining}
−cluster2	Cluster method in later iterations (default upgmb).	{upgma, upgmb, neighborjoining}
−distance1	Distance measure for iteration 1 (default kmer6_6).	{kmer6_6, kmer20_3, kmer20_4, kbit20_3, kmer4_6}
−distance2	Distance measure for later iterations (default pctid_kimura).	{pctid_kimura, pctid_log}
−gapopen	Gap opening penalty. We have used a factor for tuning the default value established by MUSCLE (default 1).	[0.8-1.2]
−gapextend	Gap extension penalty. We have used a factor for tuning the default value established by MUSCLE (default 1).	[0.8-1.2]
−hydro	Window size for determining whether a region is hydrophobic (default 5).	[2-15]
−hydrofactor	Multiplier for gap open/close penalties in hydrophobic regions (default 1.2).	[0.8-1.6]
−objscore	Objective score used by tree dependent refinement (default spm).	{sp, spf, dp, ps, xp, spm}
−root1	Method used to root tree in iteration 1 and 2 (default pseudo).	{pseudo, midlongestspan, minavgleafdist}
−root2	Method used to root tree in later iterations (default pseudo).	{pseudo, midlongestspan, minavgleafdist}
−SUEFF	Constant used in UPGMB clustering (default 0.1).	[0-0.25]
−weight1	Sequence weighting scheme in iteration 1 and 2 (default clustalw)	{none, henikoff, henikoffpb, clustalw, threeway}
−weight2	Sequence weighting scheme in later iterations (default clustalw)	{none, henikoff, henikoffpb, clustalw, threeway}
−center	Center parameter. We have used a factor for tuning the default value established by MUSCLE (default 1)	[0.8-1.2]

as default. It is clear that the framework will be slower than the aligner, but it will not be a problem if the obtained improvement is worthy. The closest parameter configuration in terms of characteristics A, B, and C simultaneously was tested, obtaining low-quality alignments; therefore, it was not included in the framework.

The PSO algorithm was written in C and the characteristic-based framework consists of a combination of C++ programs and shell-scripts. The C++ implementations were run using g++ (GCC) 4.4.7 and the machine used was a 2.3 GHz Intel PC with 1GB RAM (Scientific Linux 6.6).

TABLE II
COMPARISON BETWEEN THE DEFAULT PARAMETER CONFIGURATION OF MUSCLE AND THE BEST PARAMETER CONFIGURATION FOUND BY THE PSO ALGORITHM. NOTE THAT, THE AVERAGE ACCURACY (*Q*) AND CONSERVATION (TC) INDICATORS ARE IN PERCENTAGE (%), AND THE CPU TIME IS PRESENTED IN SECONDS

	Default			PSO Configuration		
	Q	TC	CPU Time (s)	Q	TC	CPU Time (s)
RV11	57.15	32.06	25	76.78	57.49	10
RV12	91.54	80.90	36	96.09	90.40	22
RV20	88.89	35.30	265	95.04	59.88	168
RV30	81.44	41.19	298	90.73	67.34	434
RV40	86.31	45.32	584	92.93	72.81	482
RV50	83.52	46.39	138	89.95	65.34	87
	81.48	**46.86**	**1347**	**90.25**	**68.88**	**1205**

IV. RESULTS

This section is devoted to show the effectiveness of the proposed characteristic-based framework. The MUSCLE v3.8 [14] is the selected aligner for proving the performance of the proposed framework, we refer to it as fwk-MUSCLE. We have selected MUSCLE because it is a fast, accurate, and easy-to-use aligner and one of the most widely-used methods in biology. According to its website [26], MUSCLE is cited by seven new papers every day. In Table I, we list the parameters of MUSCLE optimized by the PSO algorithm, including a description and the lower and upper bound of each one.

In order to estimate the improvement of quality gained by MUSCLE using the proposed characteristic-based framework, diverse protein benchmark databases were used.

1) *Benchmark Alignments Database (BAliBASE) v3.0 [27]:* This benchmark defines a total of 218 sets of sequences (extracted from the protein data bank) that are prepared to be aligned by MSA approaches. All the sequences were extracted from the protein data bank [28]. It divides the 218 sets of sequences in six subsets according to their families and similarities: RV11 (very divergent sequences, <20% residue identity. 38 sets of sequences), RV12 (medium to divergent sequences, 20%–40%, 44 sets of sequences), RV20 (families with one or more highly divergent sequences, 41 sets of sequences), RV30 (divergent subfamilies, 30 sets of sequences), RV40 (sequences with large terminal N/C extensions, 49 sets of sequences), and RV50 (sequences with large internal insertions, 16 sets of sequences).
2) *Protein Reference Alignment Benchmark (PREFAB) v4.0 [14]:* It is a collection of 1680 alignments. In PREFAB, two unrelated protein sequences are aligned by an structural-based algorithm. Then, each of these sequences is used to query a database by position-specific iterative basic local alignment search tool [29], retrieving high related sequences. The two targets and the resulting sequences are then combined in an MSA

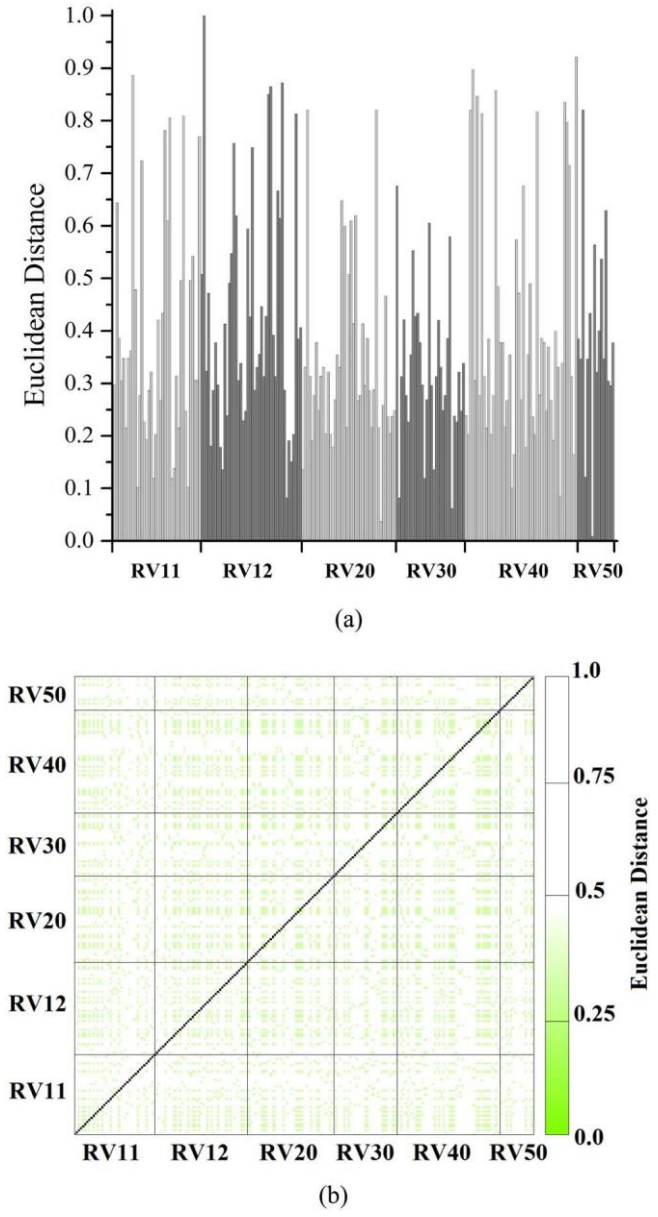

Fig. 3. Euclidean Distance comparison. (a) Euclidean distance between the default configuration of MUSCLE and each best-parameters-configuration found by PSO for the 218 datasets of BAliBASE v3.0 (stored in the characteristic-configuration file). (b) Euclidean distance between each pair of best-parameters-configurations stored in the characteristic-configuration file. Note that, given two parameters-configurations, a lower value of Euclidean distance indicates a higher level of similarity between them.

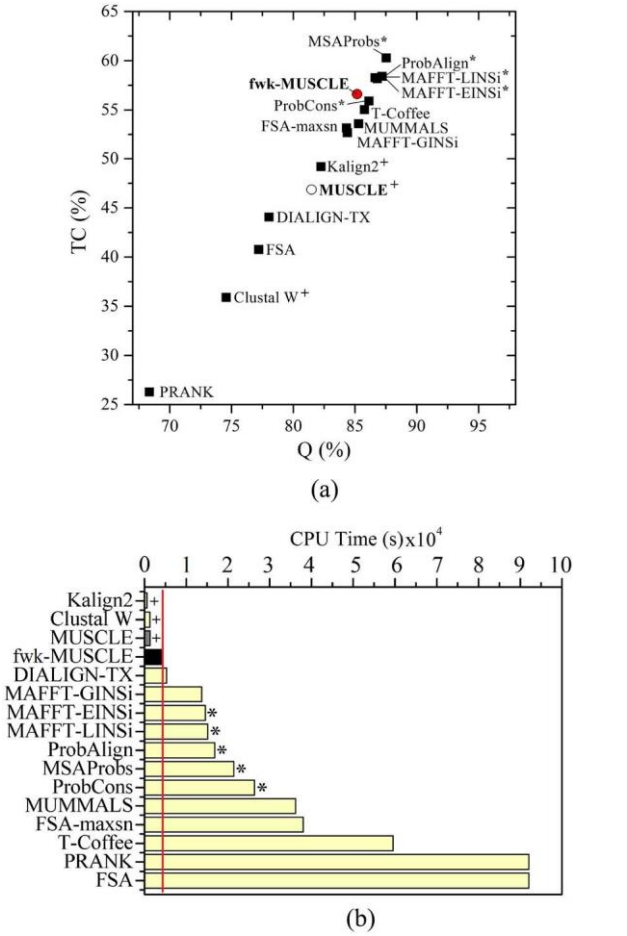

Fig. 4. Visual comparison among the different aligners in the 218 sets of BAliBASE v3.0. Note that, a "*" indicates those aligners with higher accuracy or conservation than fwk-MUSCLE, while a "+" refers to those aligners with lower CPU time (s) than fwk-MUSCLE. (a) *Q* (%) and TC (%). (b) CPU time (s).

in order to build the reference alignments. The reference quality is assessed on the original pair of sequences, by comparing with their structural alignment. In our experiments, we divide the 1680 sets in four groups according to the identity (Id): a) [0%–10%) (102 sets, where $0\% \leq Id < 10\%$); b) [10%–20%) (702 sets, where $10\% \leq Id < 20\%$); c) [20%–40%) (658 sets, where $20\% \leq Id < 40\%$); and d) [40%–100%] (218 sets, where $40\% \leq Id \leq 100\%$).

3) *Sequence Alignment Benchmark (SABmark) v1.65 [30]:* It consists of 423 sets of sequences. These sets are divided in two groups: a) Twilight (108 sets) and b) Superfamily (315 sets). On one hand, the sequence similarity in the Twilight group is very low, between 0%–25% identity. On the other hand, in the Superfamily group, the sequences share at most 50% identity.

To create the *characteristic-configuration* file, we have used the 218 sets of sequences included in the BAliBASE v3.0. The choice of using BAliBASE was based on the fact that in this benchmark the alignments are based upon protein 3-D 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 quality characteristics of BAliBASE, but it can also be considered as a source of subjectivity due to the *expert* refinement [31]. After obtaining the best parameter configuration of MUSCLE by using the procedure of Fig. 1, the results (in terms of *Q*, TC, and CPU time) obtained by the default configuration and the best configuration are presented in Table II. The parameter configurations found by PSO for the 218 sets of BAliBASE not only produces a noticeable improvement of *Q* and TC, but also the CPU time (in seconds) has been reduced.

In PSO, we optimize 20 parameters of MUSCLE (see Table I); therefore, the default configuration and each

TABLE III

BALIBASE V3.0 COMPARISON AMONG FWK-MUSCLE, DEFAULT MUSCLE [14], AND OTHER APPROACHES PUBLISHED IN THE LITERATURE. THE AVERAGE ACCURACY (*Q*) AND CONSERVATION (TC) INDICATORS ARE IN PERCENTAGE (%); THE CPU TIME INDICATES THE REQUIRED TIME TO ALIGN THE 218 DATASETS (IN SECONDS). NOTE THAT, THE ALIGNMENTS HAVE BEEN SUBDIVIDED ACCORDING TO FAMILIES: RV11, RV12, RV20, RV30, RV40, AND RV50; THE NUMBER OF MULTIPLE ALIGNMENT FILES IN EACH SUBSET IS GIVEN IN PARENTHESES

Aligner	RV11 (38)		RV12 (44)		RV20 (41)		RV30 (30)		RV40 (49)		RV50 (16)		CPU	
	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Q	TC	Time(s)	
MSAProbs	68.18	44.40	94.63	87.03	92.83	46.94	86.46	61.15	92.32	61.04	90.76	61.21	21427	*
ProbAlign	69.50	45.69	94.64	86.69	92.59	44.44	85.30	56.95	92.21	61.23	88.86	55.52	16841	*
MAFFT-LINSi	67.12	45.00	93.63	84.23	92.62	45.74	85.55	57.33	91.91	60.09	89.99	56.61	15149	*
MAFFT-EINSi	66.00	44.02	93.61	83.90	92.64	45.15	86.12	59.20	91.43	57.51	89.91	59.85	14577	*
ProbCons	66.97	41.96	94.12	86.05	91.68	41.16	84.53	54.73	90.03	53.61	89.41	57.89	26327	*
fwk-MUSCLE †	**64.77**	**43.94**	**93.10**	**85.10**	**91.21**	**41.76**	**84.73**	**54.76**	**90.27**	**60.80**	**86.90**	**53.34**	**4294**	
T-Coffee	65.72	41.41	94.47	85.93	91.57	40.60	83.69	47.76	89.64	55.38	89.49	59.11	59539	
MUMMALS	66.94	41.97	94.30	84.47	90.62	43.13	84.79	49.81	87.14	48.84	87.91	53.29	36215	
FSA-maxsn	61.88	36.58	93.65	84.79	90.10	34.34	81.37	48.26	91.61	58.46	87.19	56.57	38039	
MAFFT-GINSi	60.71	34.66	92.70	82.52	90.50	39.13	85.31	53.22	88.64	51.56	88.39	54.99	13727	
Kalign2	60.53	36.87	91.21	79.34	90.08	36.25	81.26	47.99	88.33	50.78	82.01	43.96	63	+
MUSCLE	**57.15**	**32.06**	**91.54**	**80.90**	**88.89**	**35.30**	**81.44**	**41.19**	**86.31**	**45.32**	**83.52**	**46.39**	**1347**	+
DIALIGN-TX	50.47	26.81	88.21	75.69	87.81	30.78	76.14	38.90	83.40	45.17	82.15	47.05	5292	
FSA	50.27	27.23	92.38	82.22	86.50	18.99	68.96	26.29	86.09	47.80	78.95	42.06	92056	
Clustal W	50.06	22.99	86.49	71.70	85.20	22.16	72.50	27.59	78.94	39.82	74.24	31.16	1287	+
PRANK	46.18	21.62	83.77	62.08	80.14	12.42	57.84	6.38	74.77	34.22	67.35	20.99	92050	

* aligners with higher accuracy or conservation than fwk-MUSCLE
\+ aligners with lower CPU time (s) than fwk-MUSCLE

†Note that, in 161 out of 218 datasets (73.85%), fwk-MUSCLE obtains better accuracy and consistency than default MUSCLE.

TABLE IV

PREFAB V4.0 COMPARISON AMONG FWK-MUSCLE, DEFAULT MUSCLE [14], AND OTHER APPROACHES PUBLISHED IN THE LITERATURE. THE AVERAGE ACCURACY (*Q* AND TC ARE IDENTICAL) INDICATOR IS IN PERCENTAGE (%); THE CPU TIME INDICATES THE REQUIRED TIME TO ALIGN THE 1680 DATASETS (IN SECONDS). NOTE THAT, THE ALIGNMENTS HAVE BEEN SUBDIVIDED ACCORDING TO SEQUENCE IDENTITY RANGES: [0%–10%), [10%–20%), [20%–40%), AND [40%–100%]; THE NUMBER OF MULTIPLE ALIGNMENT FILES IN EACH SUBSET IS GIVEN IN PARENTHESES

Aligner	[0-10%) (102)	[10-20%) (702)	[20-40%) (658)	[40-100%] (218)	CPU Time (s)	
MUMMALS	28.90	59.74	85.72	95.90	73400	*
MAFFT-LINSi	25.61	59.14	85.27	96.93	16923	*
ProbCons	24.37	58.15	85.33	96.11	105083	*
ProbAlign	22.76	58.30	85.85	96.77	62904	*
fwk-MUSCLE †	**25.77**	**55.54**	**83.98**	**94.84**	**13462**	
MSAProbs	22.76	56.42	84.89	94.07	88706	
MAFFT-GINSi	22.36	53.26	83.34	93.93	17837	
MAFFT-EINSi	21.28	54.42	83.19	93.84	19938	
T-Coffee	20.06	53.47	82.81	93.85	258166	
FSA-maxsn	13.60	47.98	83.07	96.76	173983	+
MUSCLE	**19.31**	**48.51**	**80.29**	**93.21**	**3761**	+
Kalign2	17.33	44.85	77.33	93.76	185	+
Clustal W	17.93	43.34	77.08	94.85	5744	
DIALIGN-TX	13.55	40.88	75.90	93.94	32269	
PRANK	7.85	31.35	65.81	90.16	861487	
FSA	3.34	24.90	72.56	93.49	394087	

* aligners with higher accuracy than fwk-MUSCLE
\+ aligners with lower CPU time (s) than fwk-MUSCLE

†Note that, in 1177 out of 1680 datasets (70.06%), fwk-MUSCLE obtains better accuracy and consistency than default MUSCLE.

TABLE V

SABMARK V1.65 COMPARISON AMONG FWK-MUSCLE, DEFAULT MUSCLE [14], AND OTHER APPROACHES PUBLISHED IN THE LITERATURE. THE AVERAGE ACCURACY (*Q*) AND CONSERVATION (TC) INDICATORS ARE IN PERCENTAGE (%); THE CPU TIME INDICATES THE REQUIRED TIME TO ALIGN THE 423 DATASETS (IN SECONDS). NOTE THAT, THE ALIGNMENTS HAVE BEEN SUBDIVIDED ACCORDING TO FAMILIES: TWILIGHT AND SUPERFAMILY; THE NUMBER OF MULTIPLE ALIGNMENT FILES IN EACH SUBSET IS GIVEN IN PARENTHESES

Aligner	Twilight (108)		Superfamily (315)		CPU	
	Q	TC	Q	TC	Time (s)	
MUMMALS	44.82	24.47	68.08	48.56	1125	*
T-Coffee	42.50	24.33	65.73	45.76	829	*
MSAProbs	42.85	22.81	66.21	45.95	237	*
fwk-MUSCLE †	**41.30**	**23.49**	**65.88**	**45.47**	**183**	
ProbCons	42.55	22.35	65.52	44.93	307	
ProbAlign	42.43	22.64	65.40	44.11	171	+
MAFFT-LINSi	39.29	18.91	64.24	42.99	518	
MAFFT-GINSi	37.95	19.53	63.06	40.86	530	
MAFFT-EINSi	37.66	18.15	63.11	42.22	582	
MUSCLE	**34.70**	**16.96**	**61.30**	**39.13**	**75**	+
FSA-maxsn	34.13	15.53	60.72	39.24	892	
Kalign2	33.74	18.14	58.43	38.39	7	+
Clustal W	31.49	15.14	58.84	37.21	24	+
DIALIGN-TX	29.86	12.29	57.11	34.94	167	+
PRANK	29.82	12.71	55.53	33.15	12036	
FSA	24.82	10.40	53.12	31.15	995	

* aligners with higher accuracy or conservation than fwk-MUSCLE
\+ aligners with lower CPU time (s) than fwk-MUSCLE

†Note that, in 302 out of 423 datasets (71.39%), fwk-MUSCLE obtains better accuracy and consistency than default MUSCLE.

best-parameters-configuration stored in the characteristic-configuration file represent a tuple with 20 values. We have measured the Euclidean distance between the default configuration tuple of MUSCLE and each best-parameters-configuration tuple found by PSO for the 218 datasets of BAliBASE v3.0. As we can see in Fig. 3(a), in almost all the datasets of BAliBASE, the best-parameters-configuration found by PSO is different from the default parameter configuration of MUSCLE. On the other hand, we have also measured the Euclidean distance between each pair of best-parameters-configuration tuples stored in the characteristic-configuration file. As we can see in Fig. 3(b), the best-parameters-configurations are also different among them.

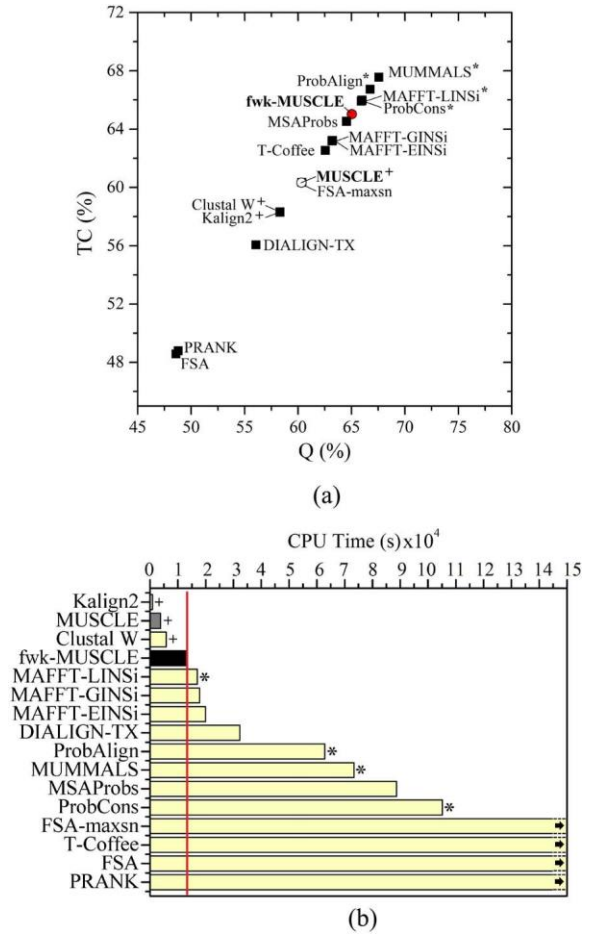

Fig. 5. PREFAB v4.0 visual comparison among the different aligners. Note that, a * indicates those aligners with higher accuracy than fwk-MUSCLE, while a # refers to those aligners with lower CPU time (s) than fwk-MUSCLE. (a) *Q* (%) and TC(%). (b) CPU time (s).

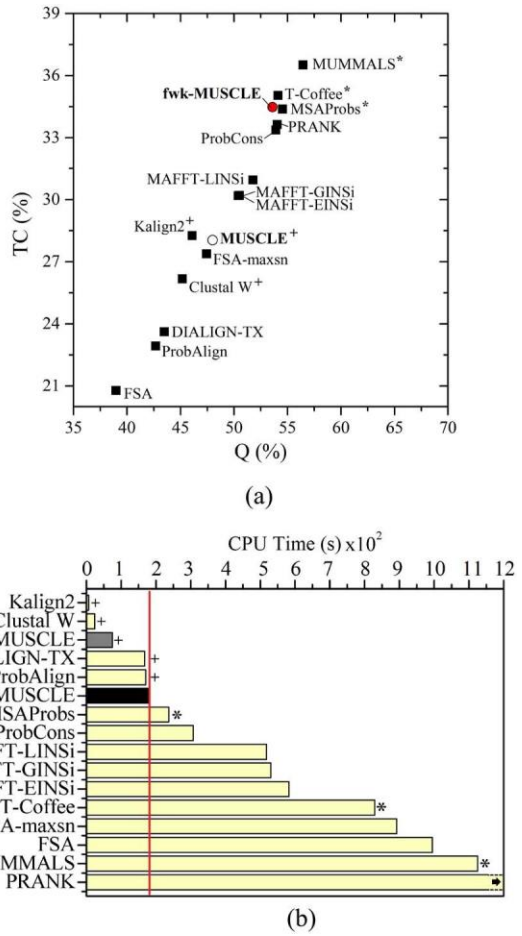

Fig. 6. SABmark v1.65 visual comparison among the different aligners. Note that, a * indicates those aligners with higher accuracy than fwk-MUSCLE, while a # refers to those aligners with lower CPU time (s) than fwk-MUSCLE. (a) *Q* (%) and TC (%). (b) CPU time (s).

The required running time of PSO for obtaining the *characteristic-configuration* file was 612 243 s. As we can see, the running time for obtaining the *characteristic-configuration* file is high; however, the parameter optimization of MUSCLE is already done (the users will not suffer this initial overhead) and a *characteristic-configuration* file is freely available[1] for potential users of the characteristic-based framework.

The *characteristic-configuration* file was created by using BAliBASE; therefore, a total of 218 independent best-parameters-configurations are stored in the configuration file. To fairly validate the proposed characteristic-based framework in the BAliBASE benchmark, we have employed the *leaving 1-out* cross validation strategy.

In order to prove the effectiveness of the proposed characteristic-based framework, we compare the default MUSCLE, the characteristic-based MUSCLE (fwk-MUSCLE), and other well-known approaches published in the literature. The approaches involved in the study are: Clustal W [6] (v2.1), DIALIGN-TX [10] (v1.0.2), FSA and FSA with the -maxn option [8] (v1.15.9), Kalign2 [9] (v2.03), MAFFT [15] (v7.215, three versions with the accurate flags: L-INS-i, E-INS-i, and G-INS-i), MSAProbs [13] (v0.9.7), MUMMALS [16] (version dated on 08/02/2008), ProbCons [12] (v1.12), PRANK [7] (with accurate option flag +F and using Clustal W tree), ProbAlign [32] (v1.4), and T-Coffee [11] (v11.0).

In the first place, we compare the *Q*, TC, and CPU time obtained by the different approaches on the BAliBASE v3.0 benchmark. In Table III, we present the results obtained by each method. As we can see, the fwk-MUSCLE clearly obtains better accuracy and conservation than the default MUSCLE in the six groups of BAliBASE, highlighting the particular performance on the RV11 group. We can observe that in 161 out of 218 datasets (73.85%), fwk-MUSCLE obtains better accuracy and consistency than default MUSCLE.

In addition, we can observe that MSAProbs, ProbAlign, MAFFT-LINSi, MAFFT-EINSi, and ProbCons obtain similar accuracy and consistency to fwk-MUSCLE in BAliBASE benchmark [see Fig. 4(a)]. However, we notice that the CPU time of fwk-MUSCLE (4294 s) is much lower than the CPU time required by the other approaches: 21427 s (MSAProbs), 16841 s (ProbAlign), 15149 s (MAFFT-LINSi), 14577 s (MAFFT-EINSi), and 26327 s (ProbCons); [see Fig. 4(b)].

[1] arco.unex.es/arl/fwk-msa/cbf-msa.zip

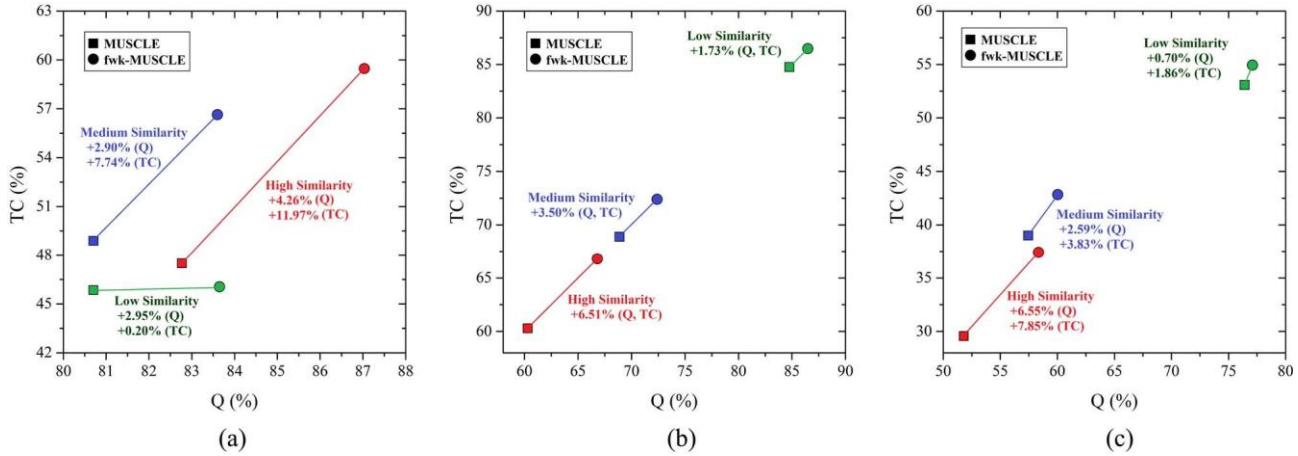

Fig. 7. Comparison between MUSCLE and fwk-MUSCLE in terms of similarity in the three benchmarks (BAliBASE v3.0, PREFAB v4.0, and SABmark v1.65). We have considered three levels of similarity of the tested dataset features: high (red), medium (blue), and low (green). (a) BAliBASE v3.0. (b) PREFAB v4.0. (c) SABMark v1.65.

Note that, the fwk-MUSCLE obtains competitive alignment results around four times faster than MSAProbs, ProbAlign, MAFFT-LINSi, MAFFT-EINSi, and ProbCons (in average).

On the other hand, we can see in Fig. 4(b) that only three approaches published in the literature present a lower CPU time than fwk-MUSCLE, they are: 1) Kalign2; 2) Clustal W; and 3) default MUSCLE. However, if we compare their alignment performance, we can notice an improvement higher than 5% in terms of *Q*, and higher than 12% in terms of TC (in average) [see Fig. 4(a)].

In the second place, we present a comparison among the different approaches on the PREFAB v4.0 benchmark. According to the developers of PREFAB v4.0 [14], the TC score is not applicable to PREFAB as the reference alignments are pairwise; therefore, we only compare the approaches by using the *Q* score and CPU time. In Table IV, we can see that fwk-MUSCLE is more accurate than default MUSCLE in all the sets of sequences included in the PREFAB benchmark. If we focus on those sets of sequences with a low percentage of identity (<20%, a total of 804 sets of sequences), we observe that fwk-MUSCLE improves the alignment accuracy in around 6%–7%.

If we compare the fwk-MUSCLE with other approaches, we observe that MUMMALS, MAFFT-LINSi, ProbCons, and ProbAlign obtain better accuracy than fwk-MUSCLE [see Fig. 5(a)]. However, in Fig. 5(b), MUMMALS, MAFFT-LINSi, ProbCons, and ProbAlign require an expensive amount of CPU time to solve the 1680 sets of PREFAB. The average runtime of the approaches that obtain better results than fwk-MUSCLE is around 65 000 s in comparison with the 13 462 s of fwk-MUSCLE, that is to say, fwk-MUSCLE is around five times faster.

In Fig. 5(b), we observe that again Kalign2, Clustal W, and default MUSCLE are faster than fwk-MUSCLE. Nevertheless, the alignment accuracy of fwk-MUSCLE is worthy. In Fig. 5(a), we can see an average accuracy improvement around 6% between fwk-MUSCLE and the other methods (Kalign2, Clustal W, and default MUSCLE).

The third benchmark under study is the SABmark v1.65. A comparison in terms of *Q*, TC, and CPU time is presented in Table V. In terms of *Q* and TC, the fwk-MUSCLE clearly improves the default MUSCLE in the two subgroups of SABmark (Twilight and Superfamily). We highlight the particular performance of the fwk-MUSCLE in the Twilight subgroup (a group with a low percentage of identity, <25%), where we find 6%–7% of improvement in terms of *Q* and TC.

In comparison with other approaches published in the literature, we can see in Fig. 6(a) that only MUMMALS obtains noticeable better accuracy (*Q*) and conservation (TC) than fwk-MUSCLE. However, if we compare the required CPU time of MUMMALS and fwk-MUSCLE in Fig. 6(b), we can notice that whereas MUMMALS requires 1125 s to align the 423 sets of sequences included in SABmark, the fwk-MUSCLE only needs 183 s, that is to say, fwk-MUSCLE is more than six times faster than MUMMALS. If we compare fwk-MUSCLE with MSAProbs, T-Coffee, PRANK, or ProbCons, we can notice that all the approaches obtain similar results in terms of *Q* and TC; however, fwk-MUSCLE required the lowest amount of time.

In Fig. 6(b), we find five approaches faster than fwk-MUSCLE, they are: 1) Kalign2; 2) Clustal W; 3) default MUSCLE; 4) DIALIGN-TX; and 5) ProbAlign. However, the alignment quality of these methods is poor in comparison with the results obtained by fwk-MUSCLE. In Fig. 6(a), in terms of *Q* and TC, we find 9% of improvement (in average) between fwk-MUSCLE and the aforementioned approaches (Kalign2, Clustal W, default MUSCLE, DIALIGN-TX, and ProbAlign).

We have divided each benchmark (BAliBASE, PREFAB, and SABmark) considering three levels of similarity of the tested dataset features with respect to the features in the characteristic-configuration file: high, medium, and low. As we can see, low-similarity datasets have in general better alignment quality (*Q*) than the medium or high-similarity ones in Fig. 7(b) and (c). Nevertheless, in Fig. 7(a), high-similarity datasets present better value of *Q* than medium-similarity ones; being the *Q* improvement obtained by the characteristic-based framework in the high-similarity datasets higher (4.20%) than

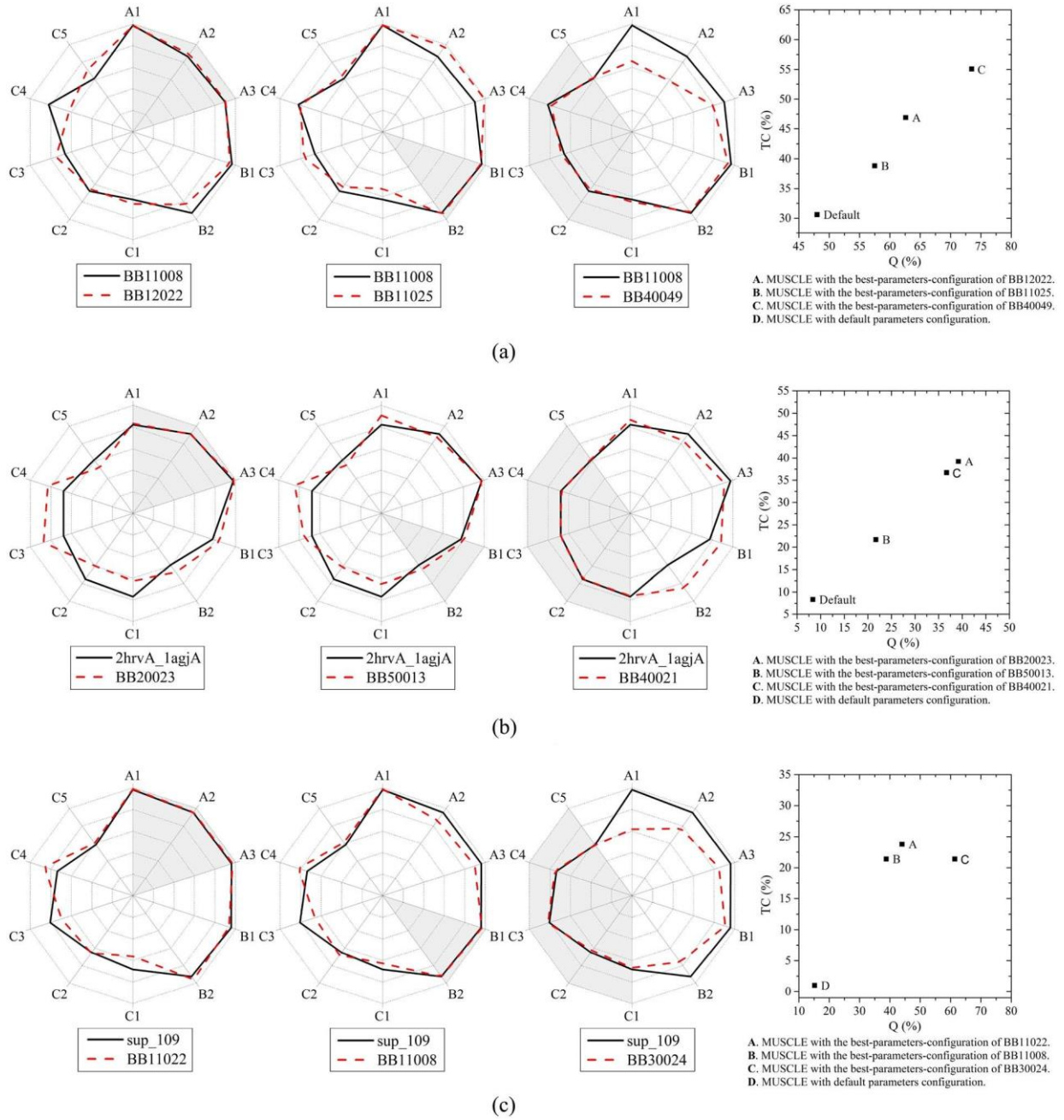

Fig. 8. Characteristic matching in BAliBASE, PREFAB, and SABmark. (a) BAliBASE v3.0 (BB11008). (b) PREFAB v4.0 (2hrvA_1agjA). (c) SABMark v1.65 (sup_109).

the *Q* improvement obtained in the medium-similarity datasets (2.90%). All in all, we can conclude that the improvement of *Q* and TC obtained by using the characteristic-based framework raises as the similarity increases.

Finally, in Fig. 8, we present an overview of the fwk-MUSCLE characteristic matching for the three bench- marks (BAliBASE, PREFAB, and SABmark). We have selected an specific set of sequences from each bench- mark: BB11008 (BAliBASE), 2hrvA_1agjA (PREFAB), and sup_109 (SABmark). For each set, we present the matching between the given data set and the most similar dataset stored in the *characteristic-configuration* file, in terms of character- istics A, B, or C. Furthermore, we compare the alignment quality between P_A, P_B, P_C, and the default parameter configuration.

After analyzing the influence of each group of characteristics in the three benchmarks, we notice that all the groups were equally useful to choose the best configuration from the *characteristic-configuration* file.

1) In 161 out of 218 datasets (73.85%) of BAliBASE v3.0, fwk-MUSCLE is better than default. From these 161 datasets: 32.92% (A), 34.16% (B), and 32.92% (C).
2) In 1177 out of 1680 datasets (70.06%) of PREFAB v4.0, fwk-MUSCLE is better than default. From these 1177 datasets: 32.54% (A), 33.14% (B), and 34.32% (C).

3) In 302 out of 423 datasets (71.39%) of SABmark v1.65, fwk-MUSCLE is better than default. From these 302 datasets: 31.46% (A), 34.44% (B), and 34.11% (C).

V. Conclusion

In this paper, a characteristic-based framework for improving the accuracy of multiple sequence aligners has been proposed. This framework analyzes the characteristics of the input set of unaligned sequences and selects a proper parameter configuration in order to improve the accuracy of the aligner.

After detailing the procedure of the framework, in the experimental phase, we apply our characteristic-based framework to one of the most widely-used aligners in the literature, MUSCLE v3.8. In order to extract some useful conclusions, different benchmark suites were used, such as BAliBASE v3.0, PREFAB v4.0, and SABMark v1.65. In this case-study, our approach was compared with the default MUSCLE and other approaches published in the literature (Clustal W, DIALIGN-TX, FSA, Kalign2, MAFFT, MSAProbs, MUMMALS, ProbCons, PRANK, ProbAlign, and T-Coffee). After analyzing the results, we conclude that the characteristic-based framework is able to greatly improve the alignment accuracy of default MUSCLE with a small penalization in runtime. Furthermore, the characteristic-based MUSCLE is able to obtain similar results than other well-known aligners, such as MUMMALS, MSAProbs, ProbCons, MAFFT, or T-Coffee, but 4–5 times faster.

A comprehensive study on the application of the proposed characteristic-based framework to other aligners published in the literature is an imminent line of future work. In addition, the use of different clustering methods in the side-chain features (group of characteristics C) will be another challenging line of future work. In this way, following the work presented in [33], an alphabet reduction procedure may be applied before computing the side-chain features of the input set of unaligned sequences. Another interesting research line would be to study in depth the biological significance of the alignments obtained by using our framework, e.g., by quantifying the number of active-sites discovered. In this paper, we use the well-known PSO in order to configure the parameters of the aligner; however, a comparative study using different approaches for tuning the parameters of the aligner is an interesting future line of work. In the literature, we find different approaches, such as the statistical method *i*-race [22], ParamILS [23], or other evolutionary approaches based on information learning like the ILABC [24] or GL-PSO, [25]. Finally, a parallel scheme of the characteristic-based framework is required in order to minimize the overhead introduced by the framework.

References

[1] D. J. Bacon and W. F. Anderson, "Multiple sequence alignment," *J. Mol. Biol.*, vol. 191, no. 2, pp. 153–161, 1986.

[2] R. F. Doolittle, "Similar amino acid sequences: Chance or common ancestry?" *Science*, vol. 214, no. 4517, pp. 149–159, 1981.

[3] D. F. Feng and R. F. Doolittle, "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *J. Mol. Evol.*, vol. 25, no. 4, pp. 351–360, 1987.

[4] C. Notredame, "Recent progress in multiple sequence alignment: A survey," *Pharmacogenomics*, vol. 3, no. 1, pp. 131–144, 2002.

[5] P. Hogeweg and B. Hesper, "The alignment of sets of sequences and the construction of phyletic trees: An integrated method," *J. Mol. Evol.*, vol. 20, no. 2, pp. 175–186, 1984.

[6] J. D. Thompson, D. G. Higgins, and T. J. Gibson, "CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice," *Nucleic Acids Res.*, vol. 22, no. 22, pp. 4673–4680, 1994.

[7] A. Löytynoja and N. Goldman, "An algorithm for progressive multiple alignment of sequences with insertions," *Proc. Nat. Acad. Sci. USA*, vol. 102, no. 30, pp. 10557–10562, 2005.

[8] R. K. Bradley *et al.*, "Fast statistical alignment," *PLoS Comput. Biol.*, vol. 5, no. 5, 2009, Art. no. e1000392.

[9] T. Lassmann, O. Frings, and E. L. L. Sonnhammer, "Kalign2: High-performance multiple alignment of protein and nucleotide sequences allowing external features," *Nucleic Acids Res.*, vol. 37, no. 3, pp. 858–865, 2009.

[10] A. R. Subramanian, M. Kaufmann, and B. Morgenstern, "DIALIGN-TX: Greedy and progressive approaches for segment-based multiple sequence alignment," *Algorithms Mol. Biol.*, vol. 3, no. 6, 2008.

[11] C. Notredame, D. G. Higgins, and J. Heringa, "T-Coffee: A novel method for fast and accurate multiple sequence alignment," *J. Mol. Biol.*, vol. 302, no. 1, pp. 205–217, 2000.

[12] C. B. Do, M. S. Mahabhashyam, M. Brudno, and S. Batzoglou, "ProbCons: Probabilistic consistency-based multiple sequence alignment," *Genome Res.*, vol. 15, no. 2, pp. 330–340, 2005.

[13] Y. Liu, B. Schmidt, and D. L. Maskell, "MSAProbs: Multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities," *Bioinformatics*, vol. 26, no. 16, pp. 1958–1964, 2010.

[14] R. C. Edgar, "MUSCLE: Multiple sequence alignment with high accuracy and high throughput," *Nucleic Acids Res.*, vol. 32, no. 5, pp. 1792–1797, 2004.

[15] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "MAFFT: A novel method for rapid multiple sequence alignment based on fast Fourier transform," *Nucleic Acids Res.*, vol. 30, no. 14, pp. 3059–3066, 2002.

[16] J. Pei and N. V. Grishin, "MUMMALS: Multiple sequence alignment improved by using hidden Markov models with local structural information," *Nucleic Acids Res.*, vol. 34, no. 16, pp. 4364–4374, 2006.

[17] H. Doğan and H. H. Otu, "Objective functions," in *Multiple Sequence Alignment Methods* (Methods in Molecular Biology), vol. 1079, D. J. Russell, Ed. New York, NY, USA: Humana Press, 2014, pp. 45–58.

[18] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Perth, WA, Australia, 1995, pp. 1942–1948.

[19] A. R. Jordehi and J. Jasni, "Particle swarm optimisation for discrete optimisation problems: A review," *Artif. Intell. Rev.*, vol. 43, no. 2, pp. 243–258, 2015.

[20] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part I: Background and development," *Natural Comput.*, vol. 6, no. 4, pp. 467–484, 2007.

[21] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Comput.*, vol. 7, no. 1, pp. 109–124, 2008.

[22] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package, iterated race for automatic algorithm configuration," IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2011-004, 2011.

[23] F. Hutter, H. H. Hoos, K. Leyton-Brown, and T. Stützle, "ParamILS: An automatic algorithm configuration framework," *J. Artif. Intell. Res.*, vol. 36, no. 1, pp. 267–306, Sep. 2009.

[24] W.-F. Gao, L.-L. Huang, S.-Y. Liu, and C. Dai, "Artificial bee colony algorithm based on information learning," *IEEE Trans. Cybern.*, vol. 45, no. 12, pp. 2827–2839, Dec. 2015.

[25] Y.-J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[26] R. Edgar. (2015). *Muscle*. [Online]. Available: http://www.drive5.com/muscle/

[27] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark," *Proteins*, vol. 61, no. 1, pp. 127–136, 2005.

[28] H. M. Berman *et al.*, "The protein data bank," *Nucleic Acids Res.*, vol. 28, no. 1, pp. 235–242, 2000.

[29] S. F. Altschul *et al.*, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," *Nucleic Acids Res.*, vol. 25, no. 17, pp. 3389–3402, 1997.
[30] I. V. Walle, I. Lasters, and L. Wyns, "SABmark—A benchmark for sequence alignment that covers the entire known fold space," *Bioinformatics*, vol. 21, no. 7, pp. 1267–1268, 2005.
[31] G. Blackshields, I. M. Wallace, M. Larkin, and D. G. Higgins, "Analysis and comparison of benchmarks for multiple sequence alignment," *In Silico Biol.*, vol. 6, no. 4, pp. 321–339, 2006.
[32] U. Roshan and D. R. Livesay, "Probalign: Multiple sequence alignment using partition function posterior probabilities," *Bioinformatics*, vol. 22, no. 22, pp. 2715–2721, 2006.
[33] J. Bacardit *et al.*, "Automated alphabet reduction for protein datasets," *BMC Bioinformat.*, vol. 10, no. 1, pp. 1–16, 2009.

Álvaro Rubio-Largo received the Ph.D. degree in computer engineering from the University of Extremadura, Caceres, Spain, in 2013.

He is currently with the NOVA Information Management School, Universidade Nova de Lisboa, Lisbon, Portugal. He has authored or co-authored over 45 publications including 19 journal citation report papers. His current research interests include parallel and distributed computing and the use of evolutionary computation for solving real-world multiobjective optimization problems in different areas, such as telecommunications and bioinformatics.

Leonardo Vanneschi received the Laurea (*summa cum laude*) degree in computer science from the University of Studies of Pisa, Pisa, Italy, in 1996, and the Ph.D. degree in computer science from the University of Lausanne, Lausanne, Switzerland, in 2004.

He is currently an Associate Professor (Tenure) with the NOVA Information Management School, Universidade Nova de Lisboa, Lisbon, Portugal. He has published about 160 scientific contributions, among which 40 have appeared in top-ranked scientific journals and ten have been honored with international awards. His current research interests include machine learning, complex systems, data mining, and evolutionary computation.

Dr. Vanneschi was a recipient of the Excellence Award of the Science Faculty of the University of Lausanne for the Ph.D. thesis. He is an Editorial Board Member of two internationally renown scientific journals. He is a Steering Committee Member and a Program Committee Member of various international conferences. He has been an Editor of several international conference proceedings and of two scientific journal special issues.

Mauro Castelli received the master's (*summa cum laude*) degree in computer science and the Ph.D. degree from the University of Milano Bicocca, Milan, Italy, in 2008 and 2012, respectively.

He is an Assistant Professor with the NOVA IMS, Universidade Nova de Lisboa, Lisbon, Portugal. His current research interests include artificial intelligence, in particular evolutionary computation and genetic programming, and in the application of machine learning techniques to solve complex real-life problems, especially in the field of biology and medicine.

Miguel A. Vega-Rodríguez received the Ph.D. degree in computer engineering from the University of Extremadura, Caceres, Spain, in 2003.

He is currently an Associate Professor (accredited as a Full Professor) of computer architecture with the Department of Computer and Communications Technologies, University of Extremadura. He has authored or co-authored over 580 publications including journal papers [over 100 journal citation report (JCR)-indexed journal papers], book chapters, and peer-reviewed conference proceedings. His current research interests include parallel and distributed computing, evolutionary computation, reconfigurable and embedded computing, and bioinformatics.

Dr. Vega-Rodríguez was a recipient of several awards, such as the ISDA'11 Best Paper Award, the IBERGRID'11 Best Paper Award, the ICEC'09 Best Paper Award, and the IEA-AIE'08 Best Paper Award. He has contributed to the organization of several international conferences and workshops, namely as the General Chair or the Co-Chair. He has edited ten special issues of international JCR-indexed journals. He is an Editor and a Reviewer of diverse international JCR-indexed journals.