A New Dynamic Programming Algorithm for Multiple Sequence Alignment

Jean-Michel Richer, Vincent Derrien and Jin-Kao Hao

LERIA - University of Angers, 2 bd Lavoisier, 49045 Angers Cedex 01, France
{richer,derrien,hao}@info.univ-angers.fr

Abstract. Multiple sequence alignment (MSA) is one of the most basic and central tasks for many studies in modern biology. In this paper, we present a new progressive alignment algorithm for this very difficult problem. Given two groups A and B of aligned sequences, this algorithm uses Dynamic Programming and the sum-of-pairs objective function to determine an optimal alignment C of A and B. The proposed algorithm has a much lower time complexity compared with a previously published algorithm for the same task [11]. Its performance is extensively assessed on the well-known BAliBase benchmarks and compared with several state-of-the-art MSA tools.

Keywords: multiple alignment, dynamic programming

1 Introduction

In biology, the *Multiple Sequence Alignment* of nucleic acids or proteins is one of the most basic and central tasks which is a prior to phylogeny reconstruction, protein structure modeling or gene annotation. The goal of the alignment operation is to identify similarities at the primary sequence level which usually implies structural and functional similarity.

A multiple alignment of a set of sequences helps visualize conserved regions of residues by organizing them into a matrix where similar residues ideally appear in the same column. In order to obtain this matrix it is necessary to use *edit operations* which consist of a match, a substitution or an insertion. A match puts two equal residues in the same column while a substitution uses two different residues. An insertion consists in inserting a special character, called a gap, whenever characters of a sequence have to be shifted from one column to be aligned with similar residues of other sequences.

To align two sequences, a simple polynomial algorithm based on dynamic programming (DP) has been designed with linear gap penalty [14]. This algorithm is based on a scoring scheme of edit operations and is influenced by two parameters: a substitution matrix and a model of gaps. A substitution matrix (PAM [1], BLOSUM [9]) assigns a score to a match or a substitution and the gap model helps score the insertions.

Obtaining an accurate alignment is a difficult task which requires to design scoring schemes which are biologically sound. In practice, the sum-of-pairs [10] is the most widely used for its simplicity although some other models have been developed [18,15].

We can compute the optimal alignment of a set of k sequences of length n by extending [14] to a k-dimension DP algorithm [13], but its complexity in $\mathcal{O}(n^k 2^k)$ is too time consuming to tackle the alignments problems that biologists encounter everyday. In fact, the problem of aligning $k > 2$ sequences is known to be NP-hard [24]. For this reason various heuristic methods have been designed to decrease the complexity of the standard algorithm and obtain sub-optimal alignments. These heuristic methods fall into two categories.

Progressive methods (PM) [5] are the most widely used optimization techniques. They consist in iteratively aligning the most closely related sequences or groups of sequences until all sequences are aligned. The most famous progressive methods are *CLUSTALW* [21] and *T-Coffee* [17], MUSCLE [4] and MAFFT [12]. The PM approach has the advantage to be simple, efficient and provides good results. Nevertheless this approach suffers from its greedy nature: mistakes made in the alignment of previous sequences can not be corrected as more sequences are added. The order in which the sequences are aligned is determined by an efficient clustering method such as neighbor-joining [19]. Progressive Methods therefore automatically construct a phylogenetic tree as well as an alignment.

The iterative methods (IM) start from an initial alignment (or a set of initial alignments) and iteratively improve it following some objective function (SAGA [16], ProbCons [3], DiAlign-T [20], PRRN/PRRP [8]). Many of these methods in fact combine iterative and progressive optimization and can lead to an alignment of better quality but generally require more computational effort [4,12].

The group-to-group (also called alignment of alignments) algorithm represents a natural simplification of the k-dimension DP algorithm and is the core of progressive and iterative methods [26]. Aligning alignments (AA) is the problem of finding an optimal alignment of two alignments under the sum-of-pairs objective function. An approximate version of AA widely used is based on profiles. A profile is a table that lists the frequencies of each amino acid for each column of an alignment. To improve the quality of the overall alignment it is interesting to compute the exact SP score of two alignments.

Recently Kececioglu and Starrett [11] gave the outline of an algorithm to exactly align two alignments with affine gap cost using shapes. We believe that this algorithm requires more computational effort than needed and can be described in a more elegant manner. We have designed a generic framework which is a generalization of the algorithm of Gotoh [6] that can be instanciated in order to perform an exact pairwise alignment or to align two alignments using linear or affine gap penalties. The method has been implemented in the software MALINBA and we give here some of the results obtained on the BAliBASE benchmarks.

The rest of the paper is organized as follows. In the next section we will give a formal description of some important notions used for the alignment of sequences. Section 3 presents the generic framework that we have defined to align alignments. The next section provides some results from the BAliBase database of alignments compared to other MSA softwares.

2 Formal definition of the problem

Let us consider that an alphabet is a set of distinct letters for which we identify a special symbol called a gap generally represented by the character `'-'`. A sequence is expressed over an alphabet and is a string of characters where each character stands for a residue, i.e. a nucleic acid (DNA) or an amino acid (protein). Aligning two sequences or two sets of sequences can be performed by using edit operations and the result in a matrix called an alignment:

Definition 1. - Alignment - *Let $S = \{S_1, \ldots, S_k\}$ be a set of sequences defined over an alphabet Σ : $\forall u \in \{1, \ldots, k\}, S_u = \langle x_1^u, \ldots, x_{|S_u|}^u \rangle$ where $|S_u|$ is the length of S_u. An alignment A^S is a matrix:*

$$A^S = \begin{bmatrix} a_1^1 \ldots a_q^1 \\ \vdots \quad\;\; \vdots \\ a_1^k \quad\;\; q_q^k \end{bmatrix}$$

such that $\forall u \in \{1, \ldots, k\}, \forall v \in \{1, \ldots, q\}, \quad a_v^u \in \Sigma$. The matrix A^S verifies the following properties:

- $\forall u \in \{1, \ldots, k\}, \quad max(|S_u|) \leq q \leq \sum_{u=1}^{u=k} |S_u|$,
- $\nexists j \in \{1, \ldots, q\}$ *such that* $\forall u \in \{1, \ldots, k\}, \quad a_j^u = -$,
- $\forall u \in \{1, \ldots, k\}$, *there exists an isomorphism* $f_u : \{1, \ldots, |S_u|\} \rightarrow \{1, \ldots, q\}$ *such that* $\langle a_{f_u(1)}^u, a_{f_u(2)}^u, \ldots, a_{f_u(|S_u|)}^u \rangle = S_u$

Example 1. For example, the set of sequences S could be aligned as follows:

S	an alignment of S
ACCT	AC-CT
AC	AC---
ACT	AC--T
CAAT	-CAAT
CT	-C--T
CAT	-CA-T

2.1 Sum-of-pairs

As previously mentioned, to establish the quality of an alignment we use an objective function called the sum-of-pairs which depends on a substitution matrix w and a model of gap $g(n)$. In the reminder of this paper we will consider that the substitution matrix corresponds to a measure of similarity which means that similar residues will be rewarded by a positive score and dissimilar residues will get a negative score. There exist two widely used models of gaps called linear and affine[1].

[1] The linear gap model is sometimes refered as a *constant model* and the affine gap model is sometimes refered as *linear* which is confusing.

Definition 2. - Gap model - *A gap model is an application $g : \mathbb{N} \rightarrow \mathbb{R}$ which assigns a score, also called a penalty, to a set of consecutive gaps. This penalty is generally negative.*

Definition 3. - Linear gap model - *For this model, the penalty is proportional to the length of the gap and is given by $g(n) = n \times g_o$ where $g_o < 0$ is the* opening *penalty of a gap and n the number of consecutive gaps.*

Definition 4. - Affine gap model - *For this model the insertion of a new gap has a more important penalty than the extension of an existing gap, this can be stated by the following formula:*

$$g(n) = \begin{cases} 0 & \textit{if} \quad n = 0 \\ g_o + (n-1) \times g_e & \textit{if} \quad n \geq 1 \end{cases}$$

where $g_o < 0$ is the gap opening penalty *and $g_e < 0$* gap extension penalty *and are such that $|g_e| < |g_o|$.*

Definition 5. - Sum-of-pairs of an alignment - *Let A^S be an alignment of a set of sequences $S = \{S_1, \ldots, S_k\}$. The sum-of-pairs is given by the following formula:*

$$sop(A^S) = \sum_{c=1}^{q} sop^c(A^S_c)$$

where $sop^c(A^S_c)$ is the score of the c column of the alignment given by:

$$sop^c(A^S_c) = \sum_{r=1}^{k-1} \sum_{s=r+1}^{k} \delta_{r,s} \times w(a^r_c, a^s_c) \times \lambda \begin{pmatrix} a^r_{c-1} \; a^r_c \\ a^s_{c-1} \; a^s_c \end{pmatrix}$$

with:

- *$0 < \delta_{r,s} \leq 1$ is a weighting coefficient that allows to remedy problems arising from biased sequences, in order for example to avoid over-represented sequences to dominate the alignment. For simplicity's sake we will chose $\delta_{r,s} = 1$ in the remainder of this paper . When $\delta_{r,s} \neq 1$ the sum-of-pairs is called the weighted sum-of-pairs [7].*
- *we introduce here λ which is the key feature of our work and is an application $\Sigma^4 \rightarrow \mathbb{R}$, induced by the gap model. λ takes into account the previous edit operation used to obtain the current column of the alignment.*

Definition 6. - λ for a linear gap model - *For a linear gap model, a gap has always the same cost wherever it is placed in the alignment. $\forall c \in \{1, \ldots, q\}$:*

$$\lambda \begin{pmatrix} a^r_{c-1} \; a^r_c \\ a^s_{c-1} \; a^s_c \end{pmatrix} = \begin{cases} 0 & \textit{if} \quad c - 1 = 0 \\ 1 & \textit{if} \quad a^r_c \neq - \quad \textit{and} \quad a^s_c \neq - \\ g_{op} & \textit{if} \quad a^r_c = - \quad \textit{or} \quad a^s_c = - \end{cases}$$

Definition 7. - λ for an affine gap model - *For the affine gap model, the previous edit operation and especially insertions will influence the cost of the penalty.* $\forall c \in \{1, \ldots, q\}$ *:*

$$\lambda \begin{pmatrix} a^r_{c-1} \; a^r_c \\ a^s_{c-1} \; a^s_c \end{pmatrix} = \begin{cases} 0 & \textit{if } c-1=0 \\ 1 & \textit{if } a^r_c \neq - \textit{ and } \quad a^s_c \neq - \\ g_{op} & \textit{if } (a^r_c = - \textit{ and } a^r_{c-1} \neq -) \textit{ or if } (a^s_c = - \textit{ et } a^s_{c-1} \neq -) \\ g_{ext} & \textit{if } (a^r_c = - \textit{ and } a^r_{c-1} = -) \textit{ or if } (a^s_c = - \textit{ et } a^s_{c-1} = -) \end{cases}$$

3 A generic framework for aligning alignments

The problem of aligning alignments can be stated as follows :

Definition 8. - Aligning alignment - *Given two multiple alignments A_v and A_h, find an optimal alignment of A_v and A_h for the sum-of-pairs objective function for a given substitution matrix w and gap model $g(n)$.*

The framework that we now define is a generalization of the algorithm of [6] based on a measure of similarity. We refer the reader to [25] for a better understanding of the computation process which is based on two steps. The first step is the initialization of the first row and first column of the matrix and the second step is the recursive relation used to compute each remaining cell of the matrix. To decrease the complexity we introduce three auxillary matrices called D, V and H. D is used to record the cost of a match or a substitution, V and H are used to record the cost of an insertion respectively in A_v and A_h. Table 1 represents the possible moves in the dynamic programming matrix. For example, DH means a match or substituion between A_v and A_h followed by an insertion in A_h. The M matrix records the optimal cost of the global alignment. To obtain the optimal alignment we use the *traceback* technique (see [25]). We consider that A_v and A_h are defined as follows :

$$A_h = \begin{bmatrix} x^1_1 & \ldots \, x^1_{q_h} \\ \vdots & \vdots \\ x^{k_h}_1 & \ldots \, x^{k_h}_{q_h} \end{bmatrix} \quad A_v = \begin{bmatrix} y^1_1 & \ldots \, y^1_{q_v} \\ \vdots & \vdots \\ y^{k_v}_1 & \ldots \, y^{k_v}_{q_v} \end{bmatrix}$$

$\searrow \downarrow \searrow \downarrow$	DD DH VD VH
$\rightarrow \searrow \rightarrow \downarrow$	DH **D** VH **V**
$\searrow \downarrow$	HD HV
$\rightarrow \rightarrow$	HH **H**

Table 1. possible moves for the dynamic programming algorithm

1. initialization, $\forall i \in \{1, \ldots, k_v\}, \forall j \in \{1, \ldots, k_h\}$:

$$\begin{array}{ll}
M_{0,0} & = D_{0,0} = H_{0,0} = V_{0,0} = 0 \\
D_{i,0} & = H_{i,0} = -\infty \\
D_{0,j} & = V_{0,j} = -\infty \\
H_{0,j} & = H_{0,j-1} + sop^{j}(A_h) + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, -) \times (g(j) - g(j-1)) \\
V_{i,0} & = V_{i-1,0} + sop^{i}(A_v) + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(-, y_i^e) \times (g(i) - g(i-1)
\end{array}$$

2. recursive relation, $\forall i \in \{1, \ldots, k_v\}, \forall j \in \{1, \ldots, k_h\}$:

$$M_{i,j} = max\{D_{i,j}, H_{i,j}, V_{i,j}\}$$

with

$$D_{i,j} = max \left\{ \begin{array}{l}
D_{i-1,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, y_i^e) \times \lambda \begin{pmatrix} x_{j-1}^f & x_j^f \\ y_{i-1}^e & y_i^e \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} y_{i-1}^r & y_i^r \\ y_{i-1}^s & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix} \\
H_{i-1,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, y_i^e) \times \lambda \begin{pmatrix} x_{j-1}^f & x_j^f \\ - & y_i^e \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} - & y_i^r \\ - & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix} \\
V_{i-1,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, y_i^e) \times \lambda \begin{pmatrix} - & x_j^f \\ y_{i-1}^e & y_i^e \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} y_{i-1}^r & y_i^r \\ y_{i-1}^s & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} - & x_j^r \\ - & x_j^s \end{pmatrix}
\end{array} \right.$$

$$H_{i,j} = max \left\{ \begin{array}{l}
D_{i,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, -) \times \lambda \begin{pmatrix} x_{j-1}^f & x_j^f \\ y_i^e & - \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(-, -) \times \lambda \begin{pmatrix} y_{i-1}^r & - \\ y_{i-1}^s & - \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix} \\
H_{i,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, -) \times \lambda \begin{pmatrix} - & x_j^f \\ - & - \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(-, -) \times \lambda \begin{pmatrix} - & - \\ - & - \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix} \\
V_{i,j-1} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, -) \times \lambda \begin{pmatrix} - & x_j^f \\ y_i^e & - \end{pmatrix} \\
+ \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(-, -) \times \lambda \begin{pmatrix} y_i^r & - \\ y_i^s & - \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} - & x_j^r \\ - & x_j^s \end{pmatrix}
\end{array} \right.$$

$$V_{i,j} = max \begin{cases} D_{i-1,j} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(-, y_i^e) \times \lambda \begin{pmatrix} x_j^f & - \\ y_{i-1}^e & y_i^e \end{pmatrix} \\ + \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} y_{i-1}^r & y_i^r \\ y_{i-1}^s & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix} \\ H_{i-1,j} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(-, y_i^e) \times \lambda \begin{pmatrix} x_j^f & - \\ - & y_i^e \end{pmatrix} \\ + \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} - & y_i^r \\ - & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(-, -) \times \lambda \begin{pmatrix} x_j^r & - \\ x_j^s & - \end{pmatrix} \\ V_{i-1,j} + \sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(-, y_i^e) \times \lambda \begin{pmatrix} - & - \\ - & y_i^e \end{pmatrix} \\ + \sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} y_{i-1}^r & y_i^r \\ y_{i-1}^s & y_i^s \end{pmatrix} + \sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(-, -) \times \lambda \begin{pmatrix} - & - \\ - & - \end{pmatrix} \end{cases}$$

The score of each edit operation depends on 4 different factors that we call α, β, γ and δ. For example, to obtain $DD_{i,j}$, we need to compute :

- the α factor which corresponds to the former edit operation $D_{i-1,j-1}$
- the β factor is the sum-of-pairs score of column j of A_h :

$$\sum_{r=1}^{k_h-1} \sum_{s=r+1}^{k_h} w(x_j^r, x_j^s) \times \lambda \begin{pmatrix} x_{j-1}^r & x_j^r \\ x_{j-1}^s & x_j^s \end{pmatrix}$$

- the γ factor is the sum-of-pairs score of the column i of A_v :

$$\sum_{r=1}^{k_v-1} \sum_{s=r+1}^{k_v} w(y_i^r, y_i^s) \times \lambda \begin{pmatrix} y_{i-1}^r & y_i^r \\ y_{i-1}^s & y_i^s \end{pmatrix}$$

- the δ factor results from the interaction of column j of A_h with column i of A_v :

$$\sum_{e=1}^{k_v} \sum_{f=1}^{k_h} w(x_j^f, y_i^e) \times \lambda \begin{pmatrix} x_{j-1}^f & x_j^f \\ y_{i-1}^e & y_i^e \end{pmatrix}$$

Proposition 1. - Complexity of aligning alignment - *The complexity of the computation of the alignment of two alignments composed of k sequences of length n is $\mathcal{O}(n^2k^2)$.*

Proof : for each $M_{i,j}$, we need to compute 9 values for which we need :

$$\underbrace{k_v \times k_h}_{\delta} + \underbrace{\frac{1}{2} \times (k_v - 1) \times k_v}_{\gamma} + \underbrace{\frac{1}{2} \times (k_h - 1) \times k_h}_{\beta}$$

computations. If we consider that $k_v = k_h = k$ and that $q_v = q_h = n$, we then have to perform : $(n+1) \times (n+1) \times 9 \times (2k-1) \times k \approx n^2 \times 9 \times 2k^2$ computations. This value is to be compared with the complexity of [11] which is $\mathcal{O}((3+\sqrt{2})^k (n-k)^2 k^{3/2})$. For example, to align 2 alignments of 10 sequences of 100 residues, [11] would normally have to perform 7.2×10^{11} computations while we would require only 1.7×10^7.

3.1 Instanciation of the framework

In the case of a pairwise alignment with a linear gap penalty, the β and γ factors are not involved because there is only one sequence in each alignment. The extension penalty is equal to the opening penalty. The simplification of formulas show that $\forall i, j\ D_{i,j} = H_{i,j} = V_{i,j}$. It is then not necessary to use the D, V and H matrices and the simplified formula is equal to the Needleman and Wunsch formula: $M_{i,j} = max\{M_{i-1,j-1} + w(x_j, y_i), M_{i,j-1} + g_o, M_{i-1,j} + g_o\}$.

4 Experimentations

The generic framework presented so far has been implemented in the software MALINBA (Multiple Affine or LINear Block Alignment) which is a modified version of PLASMA [2] written in C++. Compared to PLASMA, MALINBA uses an *exact* version of the DP algorithm to align two columns of 2 alignments while PLASMA relies on the insertion of columns of gaps in one of the two subalignments and local rearrangements of residues in blocks.

To evaluate the quality of the alignments obtained by MALINBA and other MSA programs we have performed some tests on the BAliBase 2.0 database of benchmarks.

4.1 BAliBase

BAliBase is designed for assessing MSA algorithms [22] and is divided into five reference sets which were designed to test different aspects of alignment softwares. Set 1 is composed of approximately equidistant sequences. Set 2 is made of families whith orphan sequences while Set 3 contains divergent families. Set 4 has sequences with large N/C terminal insertions and sequences of Set 5 contain large internal insertions. All reference alignments were refined manually by BAliBase authors.

To assess alignment accuracy we use the `bali_score` program which helps compute two scores : the *BAliBase sum-of-pairs score* (SPS) which in fact is a ratio between the number of correctly aligned residue pairs found in the test alignment and the total number of aligned residue pairs in core blocks of the reference alignment [23]. We also report the column score (CS) defined as the number of correctly aligned columns found in the test alignment divided by the total number of columns in core blocks of the reference alignment. The closer to 1.0 these scores are, the better the alignment is.

4.2 Results

We have compared our results obtained with MALINBA to five widely known MSA systems: (1) CLUSTALW 1.83, the most popular progressive alignment software; (2)

the nwnsi version of MAFFT 5.86 using iterative refinement techniques; (3) MUSCLE 3.6; (4) PROBCONS 1.11; (5) T-COFFEE 4.96. All sotwares were run using default parameters on an Intel Core 2 Duo E6400 with 1 Gb of RAM. For this test MALINBA used 7 specific sets of parameters and we kept the alignments that provided the best SPS scores. Given the results of table 2 which reports the average SPS end CS scores, we can rank the softwares as follows using average SPS or CS scores : CLUSTAL $<$ MALINBA, T-COFFEE $<$ MUSCLE $<$ MAFFT $<$ PROBCONS.

Softwares	Set 1		Set 2		Set 3		Set 4		Set 5		Time
	SPS	CS	SPS	CS	SPS	CS	SPS	CS	SPS	CS	(in s)
CLUSTAL	0.809	0.707	0.932	0.592	0.723	0.481	0.834	0.623	0.858	0.634	120
MAFFT	0.829	0.736	0.931	0.525	0.812	0.595	**0.947**	0.822	**0.978**	**0.911**	98
MUSCLE	0.821	0.725	0.935	0.593	0.784	0.543	0.841	0.593	0.972	0.901	75
PROBCONS	**0.849**	**0.765**	**0.943**	**0.623**	**0.817**	**0.631**	0.939	**0.828**	0.974	0.892	711
TCOFFEE	0.814	0.712	0.928	0.524	0.739	0.480	0.852	0.644	0.943	0.863	1653
MALINBA	0.811	0.705	0.911	0.522	0.752	0.346	0.899	0.734	0.942	0.842	343

Table 2. Results of the SPS and CS score for MSA Softwares and overall execution time.

Better alignments have been obtained with MALINBA by fine tuning some parameters. However these results are not reported here. Finally, let us say that despite its complexity our method is quite fast (see the time column in table 2).

5 Conclusion

We have designed a generic framework to align alignments with the sum-of-pairs objective function. This framework is exact and can be used to align two sequences or two alignments using linear of affine gap penalties. This framework was implemented in MALINBA and tested on the BAliBase benchmark and proves to be efficient. Although quite simple, we believe that many improvements can be considered in order to increase the quality of alignments obtained on the BAliBase dataset. For example, local rearrangements on misaligned regions after each progressive step using secondary structure information could probably help improve the column score.

References

1. M. O. Dayhoff, R. M. Schwartz, and B. C. Orcutt. A model of evolutionary change in proteins. In M. O. Dayhoff, editor, *Atlas of Protein Sequence and Structure*, volume 5, chapter 22, pages 345–352. National Biomedical Research Foundation, 1978.
2. V. Derrien, J-M. Richer, and J-K. Hao. Plasma : un nouvel algorithme progressif pour l'alignement multiple de séquences. *Actes des Premières Journées Francophones de Programmation par Contraintes (JFPC'05), Lens, France*, 2005.
3. C.B. Do, S.P. Mahabhashyam, M. Brudno, and S. Batzoglou. Probabilistic consistency-based multiple sequence alignment. *Genome Research*, 15:330–340, 2005.

4. R.C. Edgar. Muscle : multiple sequence alignment with high accuracy and high throughput. *Nucleic Acid Research*, 32:1792–1797, 1994.
5. D.-F. Feng and R. F. Doolitle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351–360, 1987.
6. O. Gotoh. An improved algorithm for matching biological sequences. *Journal of Molecular Biology*, Vol. 162:705–708, 1982.
7. O. Gotoh. A weighting system and algorithm for aligning many phylogenetically related sequences. *Computer Applications in the Biosciences (CABIOS)*, 11:543–551, 1995.
8. O. Gotoh. Multiple sequence alignment : algorithms and applications. *Adv. Biopys.*, 36:159–206, 1999.
9. S. Henikoff and J. G. Henikoff. Amino acid substitution matrices from protein blocks. In *Proceedings of the National Academy of Science*, volume Vol. 89, pages 10915–10919, 1992.
10. C. Humberto and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.
11. Kececioglu John and Starrett Dean. Aligning alignments exactly. In *Proceedings of RECOMB 04, March 27-31, San Diego, CA, USA*, 2004.
12. Katoh, Misawa, Kuma, and Miyata. Mafft : a novel method for rapid mulitple sequence alignment based on fast fourier transform. *Nucleic Acid Research*, 30:3059–3066, 2002.
13. D.J. Lipman, S.F. Altschul, and J.D. Kececioglu. A tool for multiple sequence alignment. In *Proc Natl Acad Sci*, pages 4412–4415, 1989.
14. S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *JMB*, 3(48):443–453, 1970.
15. C. Notredame, D. Higgins, and J. Heringa. T-coffee: A novel method for multiple sequence alignments. *Journal of Molecular Biology*, Vol. 302:205–217, 2000.
16. C. Notredame and D. G. Higgins. Saga : Sequence alignment by genetic algorithm. *Nuc. Acids Res.*, 8:1515–1524, 1996.
17. C. Notredame, D. G. Higgins, and J. Heringa. T-coffee : A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, pages 205–217, 2000.
18. C. Notredame, L. Holme, and D.G. Higgins. Coffee: A new objective function for multiple sequence alignmnent. *Bioinformatics*, Vol. 14 (5):407–422, 1998.
19. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.
20. A. R. Subramanian, J. Weyer-Menkhoff, M. Kaufmann, and B. Morgenstern. Dialign-t: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, 6:66, 2005.
21. J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustalw : improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucl. Acids Res.*, 22:4673–4690, 1994.
22. J.D. Thompson, F. Plewniak, and O. Poch. Balibase: A benchmark alignments database for the evaluation of multiple sequence alignment programs. *Bioinformatics*, Vol. 15:87–88, 1999.
23. J.D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acid Research*, 27:2682–2690, 1999.
24. L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.
25. M. S. Waterman. *Introduction to Computational Biology*. Chapman and Hall/CRC, 2000.
26. S. Yamada, O. Gotoh, and H. Yamana. Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost. *BMC Bioinformatics*, 7(1):524, 2006.