Hindawi Publishing Corporation
Journal of Artificial Evolution and Applications
Volume 2009, Article ID 963150, 10 pages
doi:10.1155/2009/963150

Research Article

Multiple Sequence Alignment Using a Genetic Algorithm and GLOCSA

Edgar D. Arenas-Díaz,[1] Helga Ochoterena,[2] and Katya Rodríguez-Vázquez[1]

[1] *Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, Universidad Nacional Autónoma de Mexico, Circuito exterior s/n, Ciudad Universitaria, 04510 Mexico, DF, Mexico*
[2] *Instituto de Biología, Universidad Nacional Autónoma de Mexico, Apdo. Postal 70-367, 04510 Mexico, DF, Mexico*

Correspondence should be addressed to Edgar D. Arenas-Díaz, xaltonalli@gmail.com

Received 14 November 2008; Revised 4 April 2009; Accepted 13 June 2009

Recommended by Jason Moore

Algorithms that minimize putative synapomorphy in an alignment cannot be directly implemented since trivial cases with concatenated sequences would be selected because they would imply a minimum number of events to be explained (e.g., a single insertion/deletion would be required to explain divergence among two sequences). Therefore, indirect measures to approach parsimony need to be implemented. In this paper, we thoroughly present a Global Criterion for Sequence Alignment (GLOCSA) that uses a scoring function to globally rate multiple alignments aiming to produce matrices that minimize the number of putative synapomorphies. We also present a Genetic Algorithm that uses GLOCSA as the objective function to produce sequence alignments refining alignments previously generated by additional existing alignment tools (we recommend MUSCLE). We show that in the example cases our GLOCSA-guided Genetic Algorithm (GGGA) does improve the GLOCSA values, resulting in alignments that imply less putative synapomorphies.

1. Introduction

The use of DNA (deoxyribonucleic acid) or protein sequences for different purposes has greatly increased as the technology for DNA and protein sequencing has improved with the consequent cost reduction. A proof for this is the enormous amount of information available in the Protein Data Bank [1] or in GenBank [2]. The exponential growth in size of these data repositories goes in parallel with the increasing need for tools to manage and analyze the valuable information therein contained. The first step to make this information manageable is to device tools to identify comparable proteins or DNA fragments, as well as comparable protein or DNA sequence units (amino acids and nucleotides, resp.). This process is referred to as sequence alignment. By aligning sequences, phylogenetic analyses can be carried out, PCR (polymerase chain reaction) primers constructed, secondary or tertiary structures predicted, among other applications. Being such a central topic, algorithms to tackle sequence alignment have already been developed. Nevertheless, as we explain more thoroughly in Section 2.2.1, sequence alignment is not a trivial problem. To reduce this complex issue to trackable problems, most available softwares consider at once pairs of sequences. Measures for alignment quality that globally use the entire data sets (matrices consisting of more than two sequences) are currently unavailable.

In this paper, we thoroughly present a Global Criterion for Sequence Alignment (GLOCSA) that uses a scoring function to globally rate multiple alignments aiming to indirectly use the parsimony criterion. We also propose an evolutionary computation technique suitable to optimize it. So this novel objective function is coupled with a Genetic Algorithm (GA), the GLOCSA-Guided Genetic Algorithm (GGGA), which uses a compact representation of the alignments and five different mutation operators to explore the solution landscape. Although GGGA can be used for completely unaligned data sets, it is more efficient for refining alignments previously generated by additional existing tools. Using GLOCSA as the scoring parameter,

GGGA is capable of improving alignments generated by other tools (MUSCLE (multiple sequence comparison by log-expectation) v3.6 [3] was used in this work to prealign the matrices).

2. Sequences and Alignments

2.1. Sequences. DNA consists of a unique sequence of repeated four nucleotides. Each nucleotide is characterized by a corresponding nitrogenous base representing the *primary structure* of a real or hypothetical DNA molecule or strand, with the capacity to carry information. Such sequences analogously exist for RNA (ribonucleic acid) and proteins [4, 5].

In biochemistry, the primary structure of a biological molecule is the exact specification of its atomic composition and the chemical bonds connecting those atoms. For molecules of DNA, RNA, or proteins, the primary structure is equivalent to specify the sequence of its monomeric subunits, that is, the nucleotides or aminoacids sequence [4, 5].

2.2. Sequence Alignment. DNA sequences, RNA sequences and the protein sequences encoded change through time, evolving mainly under the action of mutation. The simplest types of mutation are point mutations, which are *substitutions* of nucleotides or aminoacids, and insertions/deletions, also known as *indels*. When one or two comparable sequences suffered insertion and/or deletion mutations, they will differ in length (i.e., they will have a different number of nucleotides or amino acids). Because these mutations are normally not observable, it is necessary to deduce where they occurred in order to identify which nucleotides or amino acids originally occupy the same position (which ones are homologous). This is the alignment process. Although this could appear trivial, it is a complex task due to the fact that a limited and a priori known number of minimum observable units exist for each aligned position (e.g., in the case of DNA only four nucleotides) and that all positions have the same potential alternative conditions (e.g., in DNA each unaligned position needs to have one of the four nucleotides). In this way, a gap (an inferred indel) in a sequence can be placed in many positions without making a big difference with respect to the comparable sequence. To align two or more sequences, they are put together in a $(S \cdot C)$ matrix, where S is the number of sequences, and C is the maximum number of residues in a sequence(positions in the alignment); the shorter sequences are filled at the end with gap codifications ("$-$") to fit the matrix perfectly. With a sequence in each line of the matrix, the process of alignment, represents the insertion of $-$ in the sequences (see Table 11). In order to choose the *best* alignment, it is considered that, in biological terms, the process of alignment has the objective to align homologous residues (having the same evolutionary origin). Assuming that evolution is parsimonious, when performing an alignment the aim is to minimize the number of evolutionary changes (events of substitutions or indels) that the alignment implies [6].

Alignments can be either pairwise, two sequences only, or multiple, more than two sequences up to an arbitrary number. For pairwise alignments *dynamic programming* algorithms have been developed to address this problem, such as Needleman-Wunsch [7] and Smith-Waterman [8] algorithms. Pairwise alignments might be regarded as special cases of multiple alignment. In practice, however, the computational complexity of aligning multiple sequences is such that the corresponding algorithms are usually not straight extensions of the pairwise approaches. Instead, multiple alignments are often constructed by repeatedly merging pairwise alignments (*progressive alignment*) [6].

2.2.1. The Number of Possible Alignments of Two Sequences. In order to define the complexity of finding an optimal alignment given an objective function, the number of possible alignments can be computed.

Having two sequences $S_1 = S_1[1]S_1[2]\cdots S_1[m]$ and $S_2 = S_2[1]S_2[2]\cdots S_2[n]$ of size m and n, respectively, $f(m,n)$ can be defined as the number of alignments that can be formed between them.

Any possible alignment of S_1 and S_2 ends in one of these specific ways [6]:

$$\left(\frac{S_1[m]}{-}\right), \left(\frac{S_1[m]}{S_2[n]}\right), \quad \text{or} \quad \left(\frac{-}{S_2[n]}\right). \tag{1}$$

That is, the last residue of S_1 aligned with a gap codification $-$, the last residue of S_1 and S_2 aligned, or the last residue of S_2 aligned with a gap codification $-$.

Considering the effect of these three possible ends on the number of alignments that can be formed out of the remaining residues in the alignment, the ending $(S_1[m]/-)$ removes one residue from S_1, $(S_1[m]/S_2[n])$ removes one residue from both sequences and $(-/S_2[n])$ removes one residue from S_2. Following this, the next recursion can be written [6]:

$$f(m,n) = f(m-1,n) + f(m-1,n-1) + f(m,n-1). \tag{2}$$

Each of the terms in the righthand side of (2) represents a possible end. In addition to this recursion, a *stop criterion* or *boundary condition* is needed:

$$f(m,0) = f(0,n) = f(0,0) = 1. \tag{3}$$

Using the recursion in (2) and the stop criterion in (3), the number of possible alignments of two sequences of equal length from 1 to 10, $m = n = 1, 2, \ldots, 10$, can be computed (Table 1), where it is obvious that the number of alignments grows exponentially as the length of the sequences increments. Then, it is straightforward to assume that, with more sequences involved in an alignment, the number of possibilities grows even faster.

3. Previous Work

3.1. Sequence Alignment and Evolutionary Computation. Evolutionary Computation (EC) has been previously used

Table 1: Number of possible alignments for two sequences; **m** and **n** are the respective sizes of two given sequences.

m, n	No. of possible alignments
1,1	3
2,2	13
3,3	63
4,4	321
5,5	1683
6,6	8989
7,7	48639
8,8	265729
9,9	1462563
10,10	8097453

in the problem of multiple sequence alignment (MSA) [5], from Genetic Algorithms [9–11] to Evolutionary Programming [12, 13]. From these applications, SAGA (sequence alignment by genetic algorithm) is considered [10] the most relevant to the topic of this paper's research.

One of the main advantages of EC is to allow a good separation between the optimization process and the evaluation criterion (objective function). It is the objective function that defines the aim of any optimization procedure and in the case of sequence alignment, it is also the objective function that summarizes the biological knowledge that is intended to be projected into the alignment.

3.1.1. Objective Functions. An alignment is considered to be *correct* if it reflects, at least in the case of DNA, the *evolutionary history* of the species of the sequences being aligned. But, at the time of assessing the quality of an alignment, such evolutionary information is not frequently available, or even more, not known. It may also be the case that aligning a set of sequences is an intermediate step to produce an evolutionary hypothesis. Hence, alternatives must be sought, and measures of sequence similarity are an useful option. It is assumed that similar sequences share the same evolutionary origin [14], as long as the level of identity is outside the twilight zone (more than 30% identity over 100 positions). Nevertheless, to assess sequence homology by similarity has also been questioned [15, 16].

Existing measures of similarity are obtained using substitutions matrices ([17] for proteins). A substitution matrix assigns a cost for each possible substitution or conservation accordingly to the probability of occurrence, computed from data analysis. In this approach insertions and deletions are penalized (gap penalty). The most common scheme for that purpose is giving a cost for gap opening and gap extension (*affine gap penalties* model), in order to favor alignments with smaller numbers of indels (each gap can be regarded as an indel event). The main disadvantage of these substitutions matrices is that they are intended to rate the similarity between two sequences at a time only, and in order to extend them to multiple sequences, it is common to find that they are scaled by adding up each pairwise similarity to obtain the score for the multiple sequence alignment [5].

Every objective function defines a mathematical optimum (or a set of them), which is not necessarily the same as the biological optimum that is sought when aligning genetic sequences. This biological optimum can be said that arises as a consequence of the evolutionary history of the sequences in the alignment. An objective function is only as good as its mathematical optimum resembles the biological one. In order to make this two optima converge, biological knowledge must be integrated to the objective function [5].

SAGA [10] was used to optimize two different objective functions. A brief description of them are given as follows.

Weighted Sums of Pairs. Weighted Sums of Pairs is the objective function used by MSA [18]. The sums-of-pairs principle associates a cost to each pair of aligned codifications in each column of an alignment (substitution cost) and another, similar cost to the gaps (gap cost). The sum of these costs yields the global cost of the alignment. Major variations involve using (1) different sets of costs for substitutions (PAM Matrices [17], BLOSUM tables [19]), (2) different schemes for the cost of gaps (*quasinatural* and *natural* [20]), and (3) different sets of weights associated with each pair of sequences due to evolutionary distance [21].

SAGA was first used to optimize the sums of pairs with quasinatural gap penalties.

COFFEE Score. COFFEE stands for *Consistency-Based Objective Function For alignment Evaluation* and is a measure of the level of consistency between multiple alignments of a set of sequences and a library of all possible pairwise alignments of the same set of sequences. Evaluation is made by comparing each pair of aligned residues observed in the multiple alignments with the list of residue pairs that constitute the library. The consistency score is equal to the number of pairs of residues that are found simultaneously in the multiple alignment and in the library, divided by the total number of pairs observed in the multiple sequence alignment [5].

The main difference between the COFFEE function and the Weighted Sum of Pairs is the use of the library instead of the substitution matrix.

4. GLOCSA—A New Objective Function

The Global Criterion for Sequence Alignment (GLOCSA) is a new proposed function to assess the quality of multiple sequence alignments of DNA. It has been build from the ground up with simplicity and a global approach in mind. By global it is understood that it rates the alignment as a whole, that is, all sequences considered simultaneously, not taking pairs of sequences to score their corresponding alignment. It also takes into account the gaps, seeking to favor parsimony.

GLOCSA is composed of three individual criteria: *Mean Column Homogeneity (MCH)*, *Reciprocal of Gap Blocks (RGB)* and *Columns Increment (CI)*. These are combined in a polynomial with a set of corresponding weights (w_{mch}, w_{rgb}, and w_{ci}). These weights are set by default to the values shown in Table 2. This default values were determined

Table 2: GLOCSA weights.

w_{mch}	$= 1000$
w_{rgb}	$= 20$
w_{ci}	$= -20$

Table 3: Nucleic acid codifications supported.

A	Adenosine
C	Cytosine
G	Guanine
T	Thymine
R	G A (puRine)
Y	T C (pYrimidine)
K	G T (Ketone)
M	A C (aMino group)
S	G C (Strong interaction)
W	A T (Weak interaction)
B	G T C (not A) (B comes after A)
D	G A T (not C) (D comes after C)
H	A C T (not G) (H comes after G)
V	G C A (not T, not U) (V comes after U)
N	A G C T (aNy)
−	Gap
?	Any base or gap

empirically, adjusting them to assign better scores to better alignments using a set of artificial examples and some real-world alignments:

$$\text{GLOCSA} = w_{\text{mch}}\text{MCH} + w_{\text{rgb}}\text{RGB} + w_{\text{ci}}\text{CI}. \tag{4}$$

The main problem faced when scoring alignments is that the exact evolutionary history of the involved sequences is never known. Theories can be stated about which alignment reflects the more plausible or probable evolutionary history (which is what produces the differences in the sequences) but certainty cannot be guaranteed.

Compared to the other schemes of sequence alignment evaluation rating them on a pair basis, such as *weighted sum of pairs* [18], GLOCSA has the advantage of rating the whole alignment at a time (with the *Mean Column Homogeneity* criterion). It also has the advantage of considering parsimony, favoring more concentrated *gaps* (with *Reciprocal of Gap Blocks*) and smaller alignment matrices (with *Columns Increment*).

At the moment it is intended to rate only multiple sequences of DNA composed of the standard IUB/IUPAC codifications for nucleic acids, shown in Table 3.

To score an alignment of multiple sequences, a matrix with C columns and S lines is considered, where C is the maximum number of positions in a sequence, and S is the number of sequences in the alignment. Initially, to perfectly fit all the sequences in the matrix, gap positions are appended ("−") at the end of the shorter sequences.

4.1. Mean Column Homogeneity. In the alignment matrix each position is represented in a column, and the column homogeneity has the purpose of rating the grade of diversity in the elements of a given position, scoring higher the more homogeneous columns.

The basic idea is that the occurrences of each of the four bases in a column are counted. A, C, G, and T are counted with a weight of 1.0 while polymorphisms are counted as an equal fraction of a unit for each base they represent (e.g., A counts 1.0 for A while R is either G or A, so it counts 0.50 for G and 0.50 for A). Gaps are also counted, with a unit for each. Using these counts the column homogeneity for each column is computed.

The count of bases and gaps are computed in wc_{jt} $\forall 0 \leq t \leq 4$, where t is the index for a base or gap which is being counted and j is the column. These weighted counts are the result of adding up to wc_{jt} the corresponding weight (shown in Table 4) for the codification of each sequence in the column. This can be expressed as,

$$wc_{jt} = \sum_i T_w(t, \text{am}(i, j)), \tag{5}$$

where $\text{am}(i, j)$ is a function that retrieves the codification in the sequence i at column j of the alignment, and the function $T_w(t, P_c)$ looks up the weight associated with the base t and the codification P_c (in this case P_c is given by $\text{am}(i, j)$) in Table 4.

After counting, the column homogeneity of a given column is computed using the following formula:

$$\text{CH}_j = \frac{\sum_{t=0}^{3} (wc_{jt})^2}{\left(\sum_{t=0}^{4} wc_{jt}\right)^2}. \tag{6}$$

It is to be noted that in the numerator of the fraction only the four bases are considered (A, C, G, and T indexed by 0, 1, 2, and 3), and in the denominator the gap (−, indexed by 4) is considered along with the bases. This is considered in order to penalize the insertion of gaps, assuming that as the gaps are not counted in the numerator but they are counted in the denominator, the column homogeneity value decreases when there are more gaps.

In the case that a position in a sequence has a ? codification, that position for that sequence is discarded (as it was not observed) for the computing of that column homogeneity value. This is because a ? implies that in that position the sequence has no information.

A special consideration is taken when all the elements in a column are gap codifications (−) in that case the column homogeneity is given a value of zero, to penalize the existence of such columns.

When the column homogeneity value for all the columns has been computed, the mean value is obtained and that is the *Mean Column Homogeneity*.

This criterion gives higher scores to more homogeneous columns, penalizing diversity of bases in a column (as shown in the examples of Table 5).

4.2. Reciprocal of Gap Blocks. The gap codifications ("−") which are contiguous are grouped into blocks, and the

Table 4: Base count weights matrix.

t		A	C	G	T	R	Y	K	M	S	W	B	D	H	V	N	–
0	A	1	0	0	0	1/2	1/2	0	1/2	0	1/2	0	1/3	1/3	1/3	1/4	0
1	C	0	1	0	0	0	0	0	1/2	1/2	0	1/3	0	1/3	1/3	1/4	0
2	G	0	0	1	0	1/2	0	1/2	0	1/2	0	1/3	1/3	0	1/3	1/4	0
3	T	0	0	0	1	0	1/2	1/2	0	0	1/2	1/3	1/3	1/3	0	1/4	0
4	–	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Table 5: Column Homogeneity evaluation examples.

	Column												
	0	1	2	3	4	5	6	7	8	9	10	11	12
seq0	A	A	A	A	A	A	A	A	A	A	–	A	A
seq1	A	A	A	A	A	A	A	A	A	A	–	–	G
seq2	A	A	A	A	A	A	A	A	A	G	–	–	–
seq3	A	A	A	A	A	A	A	A	A	G	–	–	–
seq4	A	A	A	A	A	A	A	A	G	T	–	–	–
seq5	A	A	A	A	A	A	A	A	G	T	–	–	–
seq6	A	A	A	A	A	A	A	G	T	T	–	–	–
seq7	A	A	A	A	A	A	G	G	T	C	–	–	–
seq8	A	A	–	A	G	G	T	T	C	C	–	–	–
seq9	A	–	–	G	G	T	C	T	C	C	–	–	–
CH	1.00	0.81	0.64	0.82	0.68	0.66	0.52	0.44	0.28	0.26	0.00	0.01	0.02

reciprocal of the number of gap blocks is calculated, as shown in the next equation:

$$\text{RGB} = \frac{1}{\text{GB}}, \tag{7}$$

where GB is the number of gap blocks in the alignment. If there are no gap blocks, the Reciprocal of Gap Blocks criterion is given a value of 1.0.

This criterion serves the purpose of rewarding the alignments where the gap codifications are located in a more concentrated manner, that is, where there are fewer larger blocks of gap codifications rather than more blocks of smaller length. Fewer blocks imply less evolutionary events to be explained and a more parsimonious alignment.

In Tables 6, 7, and 8 three alignments of a hypothetical set of sequences are shown. The three alignments have the same number of "–", but the example in Table 6 has them in 3 blocks, the example in Table 7 in 2 blocks, and finally the example in Table 8 in just 1 block, a difference which is noticeable in the reciprocal gap blocks criterion, and thus favoring the alignment which implies less evolutionary events (parsimony).

4.3. Columns Increment. Inserting gaps to align a set of sequences is common, and the number of columns increases. *Columns Increment* is the ratio of this augmentation, defined by

$$\text{CI} = \frac{C}{C_0} - 1, \tag{8}$$

where C is the number of columns after aligning, and C_0 the number of columns before aligning, which is equivalent to the number of nucleotides of the longest sequence.

An example of a hypothetical set of sequences for which two different alignments are shown in Tables 9 and 10. Each alignment has a different value for the *Columns Increment* criterion. A smaller alignment is preferred because a smaller matrix probably implies less evolutionary events (parsimony).

5. GGGA—a GA Using GLOCSA

Having a new objective function to evaluate the quality of multiple sequence alignments, and considering the complexity of the problem (as it is explained in Section 2.2.1), using a genetic algorithm to optimize alignments and its GLOCSA score was considered a viable option.

GGGA, *GLOCSA-Guided Genetic Algorithm* is the Genetic Algorithm implemented to optimize the GLOCSA value. GGGA is a variant of the *Simple Genetic Algorithm* where a custom representation is proposed, along with a specific mutation operator. There is no crossover operator, selection is performed by tournament, and elitism is used.

The initialization of the population is done using the mutation operator and a seed alignment which is an input to the algorithm. To produce each individual of the next generation, an individual is selected from the previous generation, using the tournament selection operator and then submitted to the mutation operator to generate the new individual (under a mutation probability).

Table 6: Alignment to exemplify the *Reciprocal of Gap Blocks* criterion. RGB = 0.33.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
seq0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
seq1	A	A	A	A	G	G	—	—	—	C	—	—	—	A	—	—	—	G	G	A	A	A	A

Table 7: Alignment to exemplify the *Reciprocal of Gap Blocks* criterion. RGB = 0.50.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
seq0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
seq1	A	A	A	A	G	G	—	—	—	—	—	—	C	A	—	—	—	G	G	A	A	A	A

5.1. Representation of Individuals. Each individual in the population represents a possible alignment. The alignment matrix (described in Section 4, e.g., in Table 11) used to rate an alignment with GLOCSA is the base for the representation of individuals. But not everything in the matrix is necessary to reconstruct any given alignment. Therefore it is processed to obtain a much more manageable representation.

Since the solution space explored by the algorithm consists of the possible alignments of a given set of sequences which do not change, the only piece of information which is necessary to represent any alignment, is the location of every gap codification. Furthermore, if gap codifications are grouped into gap blocks (groups of contiguous gap codifications), the position and size of these blocks are the only information needed to reconstruct an alignment.

If the bases in every sequence of the alignment are indexed with consecutive numbers, starting from 0 for the first base to ease its implementation, the position of the gap blocks can be determined by the base index it precedes.

Thus, the alignment can be represented by having for each sequence a list of the positions and sizes of every gap block in them, that is, each gap block represented as two nonnegative integers (position and size). As a simple illustrative example the alignment matrix of Table 11 is transformed to its corresponding representation in Table 12. In this example, the sequence 0 has only one gap block of size 2, before the *A* with index 2 (the third one), hence the list of gap blocks for this sequence only has one element which is $[2, 2]$; sequence 1 has two gap blocks $[2, 2]$, $[3, 1]$; sequence 2 has only one $[0, 2]$, the two gap codifications at the end of the sequence were appended to fit it in the alignment matrix; so there is no need to include them in the representation (trailing gaps are a consequence of the different lengths of the sequences); sequence 3 has just one gap block $[3, 3]$.

5.2. Mutation Operator and Suboperators. The mutation operator is basically in charge of changing the gap codification appearances in the alignment represented by an individual, in order to explore the solution space. It works with a mutation probability, which determines the number of expected mutations in an individual when the operator is applied to it. As it is more manageable to refer to the mutation probability in terms of the number of mutations expected per individual, as it is more informative in the context of the problem, this approach will be used in the results.

For each mutation operation five types of changes to the gap codification appearances are proposed: insertion of new gap blocks, increment of the size of a gap block, decrease of the size of a gap block, shift of positions of gap blocks and deletion of a gap block. These five types of changes are denominated *suboperators*, and the selection of which one will be applied is determined by its individual probability, dynamically adapted throughout the generations. These suboperators were selected because in the opinion of the authors they make the algorithm capable of searching the solution space in a relatively efficient way. A crossover operator (interchanging entire sequences between alignments) was also considered but was discarded in early stages because it gave no apparent advantage to the algorithm.

It is noteworthy that while performing these changes to the alignments no penalization is done other than the modification in the GLOCSA score these changes imply.

5.2.1. Insertion Suboperator. This suboperator chooses randomly a sequence and inserts a gap block in it. The size of the new gap block is also random, but with an exponential distribution with mean fitted from the gap block sizes in the seed alignment.

The size of the new gap blocks to insert is biased toward small sizes; this is because large gap blocks are not very common, but still exist.

5.2.2. Increment Suboperator. The Increment Suboperator chooses a sequence at random and an existing gap block from it, increasing in one unit its size. If the selected sequence does not have any gap block at all, this operator leaves the sequence without change.

5.2.3. Decrease Suboperator. As the previous operator, it chooses randomly a sequence and a gap block from it, whose size will decrease by one; if the size is 1 gap codification, this operator deletes the gap block totally. Again if the selected sequence does not have any gap block at all, it remains unchanged.

5.2.4. Shift Suboperator. In a sequence chosen at random, this operator selects first a gap block in it, then a position is selected randomly in that sequence; if a gap block exists in that position, the sizes of them are interchanged. If there

Table 8: Alignment to exemplify the *Reciprocal of Gap Blocks* criterion. RGB = 1.0.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
seq0	A	A	A	A	G	G	C	A	T	C	A	T	C	A	T	C	A	G	G	A	A	A	A
seq1	A	A	A	A	G	G	C	—	—	—	—	—	—	—	—	—	A	G	G	A	A	A	A

Table 9: Alignment to exemplify the *Columns Increment* criterion. In this case, the number of columns remains the same after aligning. CI = 0.

	0	1	2	3	4	5	6	7	8
seq0	A	T	C	A	T	C	A	T	C
seq1	A	T	C	A	T	C	A	T	C
seq2	A	T	C	A	T	C	A	T	C

is not a gap in the selected position, the position of the first selected gap is set to the other position. If the selected sequence does not have any gap block at all, no modification is done. This operator mixes information within a given sequence in a single alignment (individual) it does not recombine information from two individuals as a crossover operator.

5.2.5. Deletion Suboperator. This operator selects randomly a sequence and then a gap block. This gap block is completely deleted from the list of gap blocks. If no gap block exists in the selected sequence, it remains without any change.

5.2.6. Adaptation of Mutation Suboperators Probability. The probability of applying each of the subopertors is dynamically adapted as the generations pass; it is changed accordingly to their effect in the GLOCSA score of the alignments represented by the individuals.

This adaptation is done once at the end of every generation, and the procedure is as follows.

For the first generation of the genetic algorithm the five suboperators have the same probability, each with 0.20 of probability of being used. Every time the mutation operator is applied, in a *record* are stored the GLOCSA scores before and after the mutation and a vector which represents the use count of each suboperator (e.g., in Table 13).

After generating the entire new population, the *attributed difference by suboperator* (dSO) is computed by dividing the suboperators use count by the total number of mutations performed, and then multiplying it by the difference between the after and before scores of GLOCSA. This is shown in (9), where dSO_s is the *attributed difference* for a given suboperator s, sOUC_s is the use count for suboperator s, tM is the total number of mutations performed ($\mathrm{tM} = \sum \mathrm{sOUC}_s$), and aS and bS are the GLOCSA scores after and before the mutation suboperators action:

$$\mathrm{dSO}_s = \left(\frac{\mathrm{sOUC}_s}{\mathrm{tM}}\right)(\mathrm{aS} - \mathrm{bS}) \quad \forall s = \{\text{mutation suboperators}\}. \tag{9}$$

Then, the *attributed difference by suboperator* for all the records is summed up in the *total attributed difference by suboperator* (tDSO, see (10)):

$$\mathrm{tDSO}_s = \sum_R \mathrm{dSO}_s \quad \forall s = \{\text{mutation suboperators}\}. \tag{10}$$

These tDSO_s values are then normalized by dividing them by the largest absolute value of them:

$$\mathrm{tDSO}_s = \frac{\mathrm{dSO}_s}{\max(\{|\mathrm{dSO}_s| \ \forall s\})}. \tag{11}$$

Afterward, the probability (p_s) of each suboperator is added $p_S \cdot \mathrm{SCh} \cdot \mathrm{tDSO}_s$:

$$p_s = p_s + (p_S \cdot \mathrm{SCh} \cdot \mathrm{tDSO}_s), \tag{12}$$

where SCh is a constant which sets how big the steps of the adaptation are. It was set for the experiments to the value of 0.10.

Finally the values of p_s are scaled to make the sum of all the probabilities equal to 1:

$$p_s = \frac{p_s}{\sum_S p_s}. \tag{13}$$

5.3. Population Initialization. To initialize the population a given alignment is used as a starting point. The individuals of the initial generation are mutations of it, obtained by applying the mutation operator. The mutation operator is applied discarding the adaptation stage; therefore the five suboperators have the same probability while initializing the population.

6. Tests with Real Data

6.1. Test Bench. To test the ability of GGGA to optimize the GLOCSA scoring function, three multiple sequence alignment problems were proposed, which are shown in Table 16 along with relevant information. The set of sequences *exmpl17* is a subset of *exmpl19*; the two shortest sequences were eliminated, thus presumably reducing the complexity of the alignment.

6.2. GA Test Parameters. Each set of sequences was first aligned with *MUSCLE* [3], a popular progressive alignment tool. The resulting alignment was seeded as a starting point for the initialization of the population; thus the aim of the test is to see if further improvements to the alignment of MUSCLE can be performed, guided by the GLOCSA scoring function.

The genetic algorithm for all the experiments was run over 1000 generations with a population of 100, with

Table 10: Alignment to exemplify the *Columns Increment* criterion. Here, the number of columns increased to **6** after aligning. CI = 0.66.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
seq0	A	T	C	A	T	C	—	—	—	A	T	C	—	—	—
seq1	A	T	C	—	—	—	A	T	C	A	T	C	—	—	—
seq2	A	T	C	—	—	—	—	—	—	A	T	C	A	T	C

Table 11: Alignment matrix example.

Sequence #								
0	G	A	—	—	A	C	A	G
1	G	A	—	—	A	—	A	G
2	—	—	T	C	A	C	—	—
3	G	A	T	—	—	—	A	G

Table 12: Alignment matrix example—GA representation.

Sequence #	
0	[2, 2]
1	[2, 2], [3, 1]
2	[0, 2]
3	[3, 3]

Table 13: Sample records used for adaptation.

	Suboperator					GLOCSA score	
	0	1	2	3	4	Before score	After score
					...		
15	0	1	1	0	0	34.9	34.5
16	1	0	1	0	0	34.9	35.1
					...		

5 individuals of elitism. Selection is performed using a tournament of 5 individuals.

The GLOCSA Scoring function used as the objective function has the default weights defined in Table 2.

The rate of the mutation was in the range of $[0.1, 3]$ number of expected mutations with increments of 0.1. For each of this combination of values (the previously mentioned parameters and the number of expected mutations) 30 experiments were performed.

The only parameter tested within a range was the number of expected mutations. Because it was considered the most important and performing a parameter sweep across all parameters would have been too computationally expensive.

6.3. Experiments Results. Results of these experiments are shown in Figures 1, 2, and 3, using box and whiskers plots; the box has lines at the lower quartile, median, and upper quartile values; whiskers extend from each end of the box to the minimum and maximum scores obtained.

It was observed that the GLOCSA-Guided Genetic Algorithm always improved (at least slightly) the solution previously found by MUSCLE (the score of the initial alignment is the lower range of the GLOCSA Scores in the chart), and as expected the amount of improvement is strongly related with the *number of expected mutations*; lower (near zero) and higher (close and beyond three mutations per individual) numbers of expected mutations produce less improvements while values in or in the vicinity of the range of $[0.5, 1.0]$ produce the higher optimization values. This trend is certainly due to the exploration/exploitation balance, with fewer mutations there is not enough exploration, and with too many mutations there is excess exploration in detriment of exploitation.

It is important to notice that the range of the GLOCSA values in Figures 1, 2, and 3 is different for each of them. This is because GLOCSA values are relative to the alignment they are scoring, prominently the *column homogeneity*. In particular this criterion would have a value of 1000 (multiplied by its default weight) when aligning a set of copies of a single sequence (every column will score 1.0).

While the improvements for the alignments exmpl17 and exmpl19 are about the same, for exmpl29 these are bigger.

Table 14: Default Test Parameters. For each experiment these default parameters were used.

GLOCSA weights	$w_{\text{mch}} = 1000$ $w_{\text{rgb}} = 20$ $w_{\text{ci}} = -20$
Number of generations	1000
Individuals in population	100
Elite individuals	5
Individuals in tournament	5

Table 15: Test Experiments. Using the default test parameters listed in Table 14, the number of expected mutations were tested in the range of [0.1,3] with increments of 0.1, performing 30 experiments for each configuration, for each alignment in the test bench.

Alignment	No. of expected mutations	Experiments performed
exmpl19	0.1	30
	0.2	30
	⋮	⋮
	2.9	30
	3.0	30
exmpl17	0.1	30
	⋮	⋮
	3.0	30
exmpl29	0.1	30
	⋮	⋮
	3.0	30

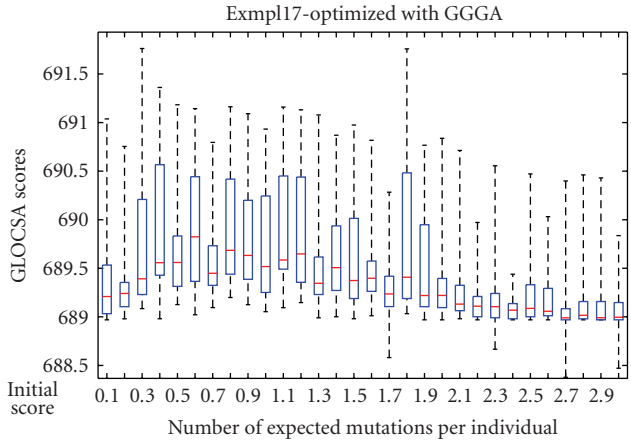

Figure 1: Box and whisker plot of the results of the experiments of exmpl17.

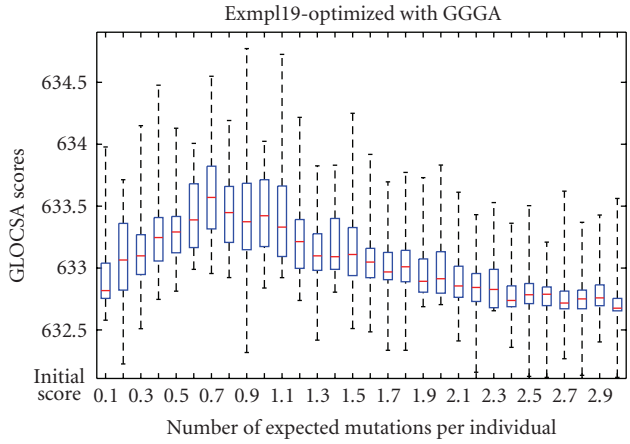

Figure 2: Box and whisker plot of the results of the experiments of exmpl19.

Table 16: Test Bench.

	No. of seq.	max. no. of pos.	Total no. of bases
exmpl19	19	649	10908
exmpl17	17	649	10149
exmpl29	29	245	6150

This is explained by the fact that exmpl29 is less complex than exmpl17 and exmpl19, which are about the same difficulty (exmpl17 easier that exmpl19, as the sequences in the first are a subset of those in the second, but not enough to make a noticeable difference).

In Table 17, the mean elapsed times for the 30 experiments of each alignment are shown. All the experiments were performed in a personal computer with an Intel Pentium D CPU 2.80 GHz processor (though not using its two cores for a single experiment) with 2 GB in RAM.

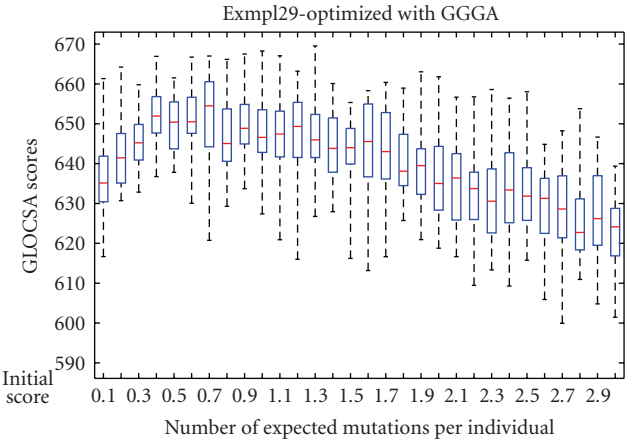

Figure 3: Box and whisker plot of the results of the experiments of exmpl29.

Table 17: Tests elapsed times.

Test	Elapsed time (minutes)
exmpl19	12.37
exmpl17	11.25
exmpl29	7.03

7. Conclusions

For the assessment of the quality of multiple sequence alignments, scoring functions have been previously defined, but in the opinion of the authors, the results obtained so far are not satisfactory enough, and therefore the GLOCSA measure was devised. It aims to be considered an alternative scoring function for multiple sequence alignments, one with the advantages of being simple, of rating the whole alignment at once, and being parsimonious.

Given the complexity of the problem of multiple sequence alignment, the techniques of Evolutionary Computation—Genetic Algorithms in particular—seem useful for optimizing this new proposed scoring function. Although it is not efficient, compared with the fast progressive alignment heuristics (e.g., MUSCLE, a run of it over the larger alignment tested in this work takes less than 4.5 seconds in the same machine) the GGGA has the ability to optimize GLOCSA as the objective function. In the light of performing it as a refinement over previously aligned data with more efficient methods (as in the test experiments where MUSCLE alignments were inserted as starting points) is a promising application.

Even though a set of sequences can be aligned from scratch optimizing its GLOCSA score with the GA, it would be too time consuming. Then, an initial starting point given by another tool seems like a good idea, in the light that progressive alignment delivers good results, but these can be further refined. The seed alignment can be the product of any alignment tool, which gives this approach additional flexibility.

8. Future Work

Currently GLOCSA only rates DNA sequence alignments; the next step would be to extend its application scope to protein sequences.

GLOCSA as a quality measure has been validated empirically, but tests to assess its reliability are still pending. This will be done with the aid of defined sets of reference alignments such as BALiBASE (protein sequence alignments) [22, 23] and the GLOCSA-Guided Genetic Algorithm, thus resulting in the assessment of both, the scoring function and the genetic algorithm implementation.

A new crossover operator (across columns) will also be implemented in the Genetic Algorithm, and its adaptation mechanism will be explored further. The performance of the Genetic Algorithm will be compared to a Random Search, to see how much the evolutionary nature of the algorithm is contributing to the results.

Acknowledgments

The authors would like to thank CONACYT under the project 61507 and SIBA-UNIBIO (UNAM) for the funding that made this work possible.

References

[1] F. C. Bernstein, T. F. Koetzle, G. J. B. Williams, et al., "The protein data bank: a computer based archival file for macromolecular structures," *Journal of Molecular Biology*, vol. 112, no. 3, pp. 535–542, 1977.

[2] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler, "Genbank," *Nucleic Acids Reseach*, vol. 34, pp. D16–D20, 2006.

[3] R. C. Edgar, "Muscle: multiple sequence alignment with high accurracy and high throughput," *Nucleic Acids Reseach*, vol. 32, no. 5, pp. 1792–1797, 2004.

[4] W. S. Klug, M. R. Cummings, and C. Spencer, *Concepts of Genetics*, Benjamin Cummings, Essex, UK, 2005.

[5] "Using genetic algorithms for pairwise and multiple sequence alignments," in *Evolutionary Computation in Bioinformatics*, G. B. Fogel and D. W. Corne, Eds., chapter 5, Morgan Kaufman, San Francisco, Calif, USA, 2003.

[6] B. Haubold and T. Wiehe, *Introduction to Computational Biology: An Evolutionary Approach*, Birkhäuser, Basel, Switzerland, 2007.

[7] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[8] T. F. Smith and M. S. Waterman, "Comparison of biosequences," *Advances in Applied Mathematics*, vol. 2, no. 4, pp. 482–489, 1981.

[9] M. Ishikawa, T. Toya, and Y. Tokoti, "Parallel iterative aligner with genetic algorithm," in *Proceedings of the 13th International Conference on Artificial Ingelligence and Genome Workshop*, pp. 84–93, 1993.

[10] C. Notredame and D. G. Higgins, "SAGA: sequence alignment by genetic algorithm," *Nucleic Acids Research*, vol. 24, no. 8, pp. 1515–1524, 1996.

[11] C. Notredame, E. A. O'Brien, and D. G. Higgins, "RAGA: RNA sequence alignment by genetic algorithm," *Nucleic Acids Research*, vol. 25, no. 22, pp. 4570–4580, 1997.

[12] K. Chellapilla and G. Fogel, "Multiple sequence alignment using evolutionary programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, p. 452, Washington, DC, USA, July 1999.

[13] L. Cai, D. Juedes, and E. Liakhovitch, "Evolutionary computation techniques for multiple sequence alignment," in *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC '00)*, vol. 2, pp. 829–835, 2000.

[14] C. Sander and R. Schneider, "Database of homology-derived protein structures and the structural meaning of sequence alignment," *Proteins: Structure, Function and Genetics*, vol. 9, no. 1, pp. 56–68, 1991.

[15] J. I. Davis and J. J. Doyle, "Homology in molecular phylogenetics: a parsimony perspective," in *Molecular Systematics of Plants II*, pp. 101–131, Kluwer Academic Publishers, Boston, Mass, USA, 1998.

[16] H. Ochoterena, "Homology in coding and non-coding DNA sequences: a parsimony perspective," *Plant Systematics and Evolution*.

[17] M. O. Dayhoff, *Atlas of Protein Sequence and Structure*, National Biomedical Research Fundation, Washington, DC, USA, 1978.

[18] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, "A tool for multiple sequence alignment," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 86, no. 12, pp. 4412–4415, 1989.

[19] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, no. 22, pp. 10915–10919, 1992.

[20] S. F. Altschul, "Gap costs for multiple sequence alignment," *Journal of Theoretical Biology*, vol. 138, no. 3, pp. 297–309, 1989.

[21] S. F. Altschul and D. J. Lipman, "Trees, stars, and multiple biological sequence alignment," *SIAM Journal on Applied Mathematics*, vol. 49, no. 1, pp. 197–209, 1989.

[22] J. D. Thompson, F. Plewniak, and O. Poch, "BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs," *Bioinformatics*, vol. 15, no. 1, pp. 87–88, 1999.

[23] A. Bahr, J. D. Thompson, J.-C. Thierry, and O. Poch, "BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations," *Nucleic Acids Research*, vol. 29, no. 1, pp. 323–326, 2001.