Aligning Multiple Protein Sequences by Hybrid Clonal Selection Algorithm with Insert-Remove-Gaps and BlockShuffling Operators

V. Cutello[1], D. Lee[2], G. Nicosia[1], M. Pavone[2], and I. Prizzi[3]

[1] Department of Mathematics and Computer Science
University of Catania
Viale A. Doria 6, 95125 Catania, Italy
{vctl, nicosia, mpavone}@dmi.unict.it
[2] IBM-KAIST Bio-Computing Research Center
Department of BioSystems, KAIST
373-1, Guseong-dong, Yuseong-gu, Daejeon, Republic of Korea
dhlee@biosoft.kaist.ac.kr, mario@biosoft.kaist.ac.kr
[3] Diogenes Research Center, Catania, Italy
prizzi@crsdiogenes.it

Abstract. Multiple sequence alignment (MSA) is one of the most important tasks in biological sequence analysis. This paper will primarily focus on on protein alignments, but most of the discussion and methodology also applies to DNA alignments. A novel hybrid clonal selection algorihm, called an aligner, is presented. It searches for a set of alignments amongst the population of candidate alignments by optimizing the classical *weighted sum of pairs* objective function. Benchmarks from BaliBASE library (v.1.0 and v.2.0) are used to validate the algorithm. Experimental results of BaliBASE v.1.0 benchmarks show that the proposed algorithm is superior to PRRP, ClustalX, SAGA, DIALIGN, PIMA, MULTIALIGN, and PILEUP8. On BaliBASE v.2.0 benchmarks the algorithm shows interesting results in terms of SP score with respect to established and leading methods, i.e. ClustalW, T-Coffee, MUSCLE, PRALINE, ProbCons, and Spem.

Keywords: bioinformatics, multiple sequence alignment, protein sequences, immune algorithms, clonal selection algorithms, hypermutation operator.

1 Introduction

Proteomics Multiple Sequence Alignment (MSA) plays a central role in molecular biology, as it can reveal the constraints imposed by structure and function on the evolution of whole protein families [1]. MSA has been used for building phylogenetic trees, identification of conserved motifs, and predicting secondary and tertiary structures for RNA and proteins [2].

In order to be able to align a set of bio-sequences a reliable objective function able to measure an alignment in terms of its biological plausibility through an analytical or computational function is needed. Alignment quality is often the limiting factor in the analysis of biological sequences — defining an appropriate and efficient objective function can remove this limitation. It is an active research field [3]. A simple objective function to optimize is the *weighted sums-of-pairs* (SP) with affine gap penalties [4], where each sequence receives a weight proportional to the amount of independent information it contains [5] and the cost of the multiple alignment is equal to the sum of the cost of all the weighted pairwise substitutions.

This research paper proposes a Hybrid Clonal Selection Algorithm (CSA) which incorporates specific perturbation operators for MSA of amino-acids sequences. The obtained results show that the proposed Immune Algorithm is comparable to state-of-art algorithms.

2 The Multiple Sequence Alignment Problem

To determine if two biological sequences have common sub-sequences is the most popular sequence analysis problem. As described in [2] there are four fundamental topics: (1.) what *kinds of alignment* should be considered; (2.) the *scoring function* adopted to evaluate alignments; (3.) the *alignment algorithm* designed to find optimal (or suboptimal) scoring alignments; (4.) the statistical methods used to assess the *significance of an alignment score*. This paper focuses on the key issues of design and efficient implementation of alignment algorithms of finding optimal and suboptimal alignments of protein structures — but the technique is also applicable to DNA alignments.

Definition 1 ***[Sequence Alignment].*** *Let $S = \{S_1, S_2, \ldots, S_n\}$ be a set of n sequences (strings) over a finite alphabet Σ, each sequence S_i consisting of ℓ_i ordered characters $s_{i,j}$:*

$$S_i = s_{i,1}s_{i,2}\ldots s_{i,\ell_i}, \quad \forall i = 1, 2, \ldots, n$$

Let $\hat{\Sigma}$ a new alphabet: $\hat{\Sigma} = \Sigma \cup \{-\}$ by adding the symbol dash '-' to represent gaps.

Then a set $\hat{S} = \{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n\}$ of sequences over the alphabet $\hat{\Sigma}$ is called a **sequence alignment** *of the set of sequence S, if the following properties are fulfilled:*

1. *All strings in $\hat{S}$ have the same length $\hat{\ell}$ with*

$$\max_{i=1\ldots n}(\ell_i) \leq \hat{\ell} \leq \sum_{i=1}^{n} \ell_i.$$

$\hat{S}$ can be interpreted as $n \times \hat{\ell}$ matrix where the $i-th$ row contains string $\hat{S}_i$.

2. *Ignoring gaps, sequence $\hat{S}_i$ is identical with sequence S_i, $\forall i = 1, 2, \ldots, n$.*
3. *$\hat{S}$ has no columns that contains gaps only.*

When $n = 2$ a *pairwise sequence alignment* is found, with $n \geq 3$ *multiple sequence alignment.* Solving the sequence alignment problem requires a *scoring function* to evaluate alignments. A simple scoring function is a distance function (another scoring function is the similarity approach). Having a distance function $d(\hat{S}_i, \hat{S}_j)$ for any aligned sequences $\hat{S}_i$ and $\hat{S}_j$, the pairwise alignment problem can be stated as follows:

Definition 2 ***[Pairwise alignment problem].*** *Let $S = \{S_1, S_2\}$ be a set of* 2 *sequences over the alphabet Σ. Compute the alignment $\hat{S} = \{\hat{S}_1, \hat{S}_2\}$ of S over the alphabet $\hat{\Sigma}$ that minimises the distance $d(\hat{S}_1, \hat{S}_2)$.*

Hence, the multiple sequence alignment problem can be stated as follows:

Definition 3 ***[Sum-of-pairs multiple alignment problem]***
Let $S = \{S_1, S_2, \ldots S_n\}$ be a set of n sequences over the alphabet Σ. Compute the alignment $\hat{S} = \{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n\}$ of S over the alphabet $\hat{\Sigma}$ that minimises the sum of the distance over all pairs $\hat{S}_i$, and $\hat{S}_j$:

$$\min_{\hat{S}} = \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(\hat{S}_i, \hat{S}_j) \right)$$

The scoring functions previously defined are too simple to be used when aligning real biological sequences. A scoring function needs to be based on the similarity of the characters occurring in the sequences, e.g. amino-acids. For instance, for two amino-acids, aa_i and aa_j, we need a measure of the probability that they have a common ancestor, or that one aa is the result of one or several mutations of the other. This measure can be formulated as follows:

Definition 4 ***[Scoring matrix].*** *Let M be a $\ell \times \ell$ scoring matrix, where ℓ is the cardinality of the alphabet Σ, which for any two characters a and b of the alphabet Σ has the following properties:*

1. $M(a, b) = M(b, a), \qquad \forall a, b \in \Sigma,$
2. $M(a, -) = GEP$, *where GEP is a fixed gap penalty,*
3. $M(-, -) = 0.$

In general a gap of lenght h has a penalty score of $h \times GEP$, where $GEP < 0$ is the fixed gap (extension) penalty. This is called the *linear gap penalty function.* From a biological point of view a more appropriate penalty score is the *affine gap penalty function,* (AGPS): given an aligned sequence $\hat{S}_i$, the first gap receives a *gap opening penalty,* $GOP < GEP < 0$, which is stronger than penalty for gap extending spaces. Hence, a gap of lenght h has a cost of $GOP + (h-1)GEP$. The most common scoring matrices are the PAM and BLOSUM series. These scoring matrices have been developed based on observed mutations in the nature. In order to minimise redundant information, based on the relatedness of the given sequences, each sequence usually receives a weight proportional to the amount of independent information it contains. This kind of information can be derived from a phylogenetic tree for the sequences.

Definition 5 ***[Weighted symbol score].*** *Let W be such a weight matrix for every pair of aligned sequences. Then the weighted symbol score for the aligned sequences $\hat{S}_i$ $\hat{S}_j$ is defined as:*

$$WSS(\hat{S}_i, \hat{S}_j) = W_{ij} \sum_{k=1}^{\hat{\ell}} M(\hat{s}_{i,k}, \hat{s}_{j,k})$$

Sequence weights can be determined by constructing a guide tree from known sequences — this is the approach used in this paper. These definitions lead to the most common faced sum-of-pairs multiple alignment problem: optimizing a weighted sum-of-pairs function with affine gap penalties.

Definition 6 ***[Sum-of-pairs multiple alignment problem]***
Let $S = \{S_1, S_2, \ldots, S_n\}$ be a set of n sequences over the alphabet Σ. Compute the alignment $\hat{S} = \{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n\}$ of S over the alphabet $\hat{\Sigma}$ that maximises the weighted symbol score and the affine gap penalty score for all aligned sequences $\hat{S}_i$:

$$\max_{\hat{S}} \left(\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} WSS(\hat{S}_i, \hat{S}_j) + \sum_{i=1}^{n} AGPS(\hat{S}_i) \right) \qquad (1)$$

For multiple protein sequence alignment, the weighted sum-of-pairs with affine gap penalties is a popular objective function included in many MSA packages. The problem of finding the multiple alignment was investigated in [6] and [7], and proved to be a NP-hard problem. However, the results presented in [7] was proved using a *not metric scoring matrix* (zero distance between two identical residues), which is different from the actual scoring matrices used in multiple alignments. Therefore, in [6], the authors improved the previous investigation using a fixed metric score matrix through a reduction from the *Minimum Vertex Cover*, a classical NP complete problem [8]. Multiple sequence alignment (MSA) decision problems can be formulated as: given a set $S = \{S_1, \ldots, S_n\}$ of sequences, a sum-of-pairs objective function, and an integer C. MSA checks for alignments of S, which have value C or less.

3 Hybrid Clonal Selection Algorithm

This work presents a Clonal Selection Algorithm (CSA) [30] with new hypermutation operators for solving the multiple sequence alignment problem. CSAs are a special class of Immune algorithms (IAs) inspired by the human Clonal Selection Principle [31]. They are effective methods for search and optimization in real-world applications. The algorithm is population based where each individual of the population is a *candidate solution* belonging to the fitness landscape of a given computational problem. It uses two different methodologies to create the initial population, as well as new hypermutation operators which insert or remove gaps in the sequences.

Gap columns which have been matched are moved to the end of the sequence. Next the remaining elements (amino acids in this work) and existing gaps are shifted into the freed space. The designed CSA considers only two immunological entities: antigens (Ags) and B cells. The Ag is the problem to solve, i.e. a given MSA instance, and B cells are the candidate solutions, i.e. a set of alignments, that have solved (or approximated) the initial problem [32,33]. Tackling the multiple sequence alignment problem Ags and B cells are represented by a sequences matrix.

Let $\Sigma = \{A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$ be the alphabet, where each symbol represents twenty amino acids and let $S = \{S_1, S_2, \ldots, S_n\}$ be the set of $n \geq 2$ sequences with length $\{\ell_1, \ell_2, \ldots, \ell_n\}$, such that $S_i \in \Sigma^*$. Therefore, an Ag is represented by a matrix of n rows and $max\{\ell_1, \ldots, \ell_n\}$ columns, whereas for the B cells a $(n \times \ell)$ matrix was used, with $\ell = (\frac{3}{2} \cdot max\{\ell_1, \ldots, \ell_n\})$. These values where taken from experimental the proposed algorithm was able to develop more *compact alignments*. In particular, for the B cells a binary matrix was used, where $s'_{i,j} = 0$ refers to a gap in the alignment and $s'_{i,j} = 1$ to a residue with $1 \leq i \leq n$ and $1 \leq j \leq \ell$.

A Initialize the Population

Two different strategies were used to create the initial population $(t = 0)$ of candidate alignments. The first strategy, *random_initialization*, is based on the use of random "*offsets*" to shift the initial sequences in the following way: an offset is randomly chosen in the range $[0, (\ell - \ell_i)]$ by a uniform distribution and then the sequence S_i is shifted from an offset positions towards the right side of the row i, of the current B cell.

A second way to initialize the population was analyized, seeding the initial population with CLUSTALW and *CLUSTALW-seeding*. However, a percentage of the population was initialized using the offsets strategy described above to avoid the algorithm getting trapped in a local optima. Hence, the second strategy creates a percentage of initial alignments using CLUSTALW and the remaining alignments are determined by a random offsets creation.

Preliminary experimental results show that the proposed algorithm achieves better performance using the second strategy. Therefore, all results shown in this paper were obtained using a combination of the two previously introduced strategies (80% of B cell population by CLUSTALW seeding and 20% of B cell population by random_initialization using the random offsets).

The presented hybrid IA incorporates the classical *static cloning operator*, which clones each B cell *dup* times producing an intermediate population $P^{(clo)}_{N_c}$ of $N_c = d \times dup$ B cells, where d is the population size).

The basic mutation processes which are considered in pairwise alignment and multiple sequence alignments are: *substitutions* which change sequences of amino acids, as well as *insertions* and *deletions* which add or remove amino acids and/or gaps. In a first version of the algorithm the classical hypermutation and hypermacromutation operators where used: first operator flips a bit, using a number of mutations inversely proportional to the fitness function value [34], whereas the hypermacromutation simply swaps two randomly choosen subsequences. However, the first experiments produced non optimal alignments obtained, leading

Table 1. Pseudo-code of the proposed hybrid immune algorithm for the MSA

```
**Hybrid Immune Algorithm**$(d, dup, \tau_B, T_{max})$
1.  $t \leftarrow 0$;
2.  $FFE \leftarrow 0$;
3.  $N_c \leftarrow d \times dup$;
4.  $P_d^{(t)} \leftarrow$ Initialize_Population($d$);
5.  Strip_Gaps($P_d^{(t)}$);
6.  Evaluate($P_d^{(t)}$);
7.  $FFE \leftarrow FFE + d$;
8.  **while** $(FFE < T_{max})$**do**
9.      $P_{N_c}^{(clo)} \leftarrow$ Cloning $(P_d^{(t)}, dup)$;
10.     $P_{N_c}^{(gap)} \leftarrow$ Gap_operators $(P_d^{(clo)})$;
11.     Strip_Gaps($P_{N_c}^{(gap)}$);
12.     Evaluate($P_{N_c}^{(gap)}$);
13.     $FFE \leftarrow FFE + N_c$;
14.     $P_{N_c}^{(block)} \leftarrow$ BlockShuffling_operators $(P_d^{(clo)})$;
15.     Compute_Weights();
16.     Normalize_Weights();
17.     Strip_Gaps($P_{N_c}^{(block)}$);
18.     Evaluate($P_{N_c}^{(block)}$);
19.     $FFE \leftarrow FFE + N_c$;
20.     $({}^aP_d^{(t)}, {}^aP_{N_c}^{(gap)}, {}^aP_{N_c}^{(block)}) =$ Elitist-Aging$(P_d^{(t)}, P_{N_c}^{(gap)}, P_{N_c}^{(block)}, \tau_B)$;
21.     $P_d^{(t+1)} \leftarrow (\mu + \lambda)$-Selection$({}^aP_d^{(t)}, {}^a P_{N_c}^{(gap)}, {}^a P_{N_c}^{(block)})$;
22.     $t \leftarrow t + 1$;
23. **end_while**
```

to frequent premature convergence to the local optimal during the convergence process. Therefore, we have developed new hypermutation operators, specific to the multiple sequence alignments, which insert or remove gaps in the sequences — called *GAP operator* or *BlockShuffling operator*.

B GAP Operator

This operator acts on the cloned B cells generating a new population $P_{N_c}^{(gap)}$. It is based on two procedures, one inserts (InsGap), and the other removes (RemGap) adjacent sequences of gaps. Initially, the GAP operator chooses what procedure to apply using a random uniform distribution, i.e. if a number of adjacent gaps needs to be inserted into the sequences or removed. Then a number k, in the range $[1, \theta]$, of (adjacent) gaps is randomly choosen, where θ represents a percentage of the alignments length. After several experiments setting $\theta = 2\%$ was obtained.

The InsGap procedure can be summarize in the following steps: split the n sequences in z groups. During the experimental tests, $z = 2$ has been the best setting for the performances of the proposed algorithm. Hence, we can rephrase this step as follows: randomly choose a value $m \in [1, n[$, and split the n sequences into two groups: $1st$ group from 1 to m sequences, and $2nd$ group from $(m+1)$ to n; randomly choose two integer values x and y, in such way that k adjacent gaps are insterted beginning from column x for the first group, and from column y for the second group; randomly choose a subsequence shift direction D, either left or right; finally, to insert the k adjacent gaps in the relative positions for each sequence, and shift the subsequence to the D direction. During the shifting phase, it is possible to miss $n \geq 0$ bits with value 1; in this case, InsGap will select n bits with value 0, different from the k gaps inserted, and they will be flipped to 1, rebuilding the correct sequence. Figure 1, plot (a), shows an example of how the InsGap procedure works, with $k = 3$, $m = 2$ and right shift direction.

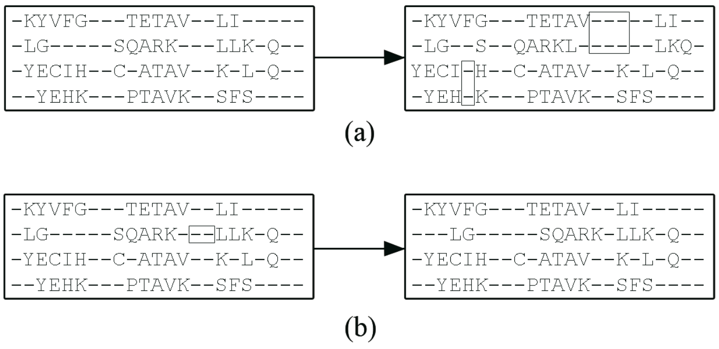

Fig. 1. GAP operator has the purpose to insert, by *InsGap* procedure (a), or remove, by *RemGap* procedure (b), adjacent gaps into the proposed alignment

RemGap procedure, simply, remove k adjacent gaps, and move the subsequences towards a randomly chosen direction, either left or right. Plot (b) of figure 1 shows an example of such an operator.

C BlockShuffling Operator

The second perturbation operator is the BlockShuffling operator, which is based on the block definition. This operator moves aligned blocks left or right: a block is selected in each alignment starting from a random point in a sequence.Three different approaches where developed: BlockMove moves whole blocks either to the left or to the right; BlockSplitHor divides the blocks in two levels, upper and lower, and shifting only one level, randomly chosen; and BlockSplitVer, which randomly choose a column in the block which divides it into two sides (left and right), and shifting only one side, randomly chosen as well. Figure 2 summarizes the three operators.

Finally, the function Strip_Gaps($P^{(*)}$) moves matched gap columns to the right end side of the matrix. This function is always applied before the fitness function is evaluated.

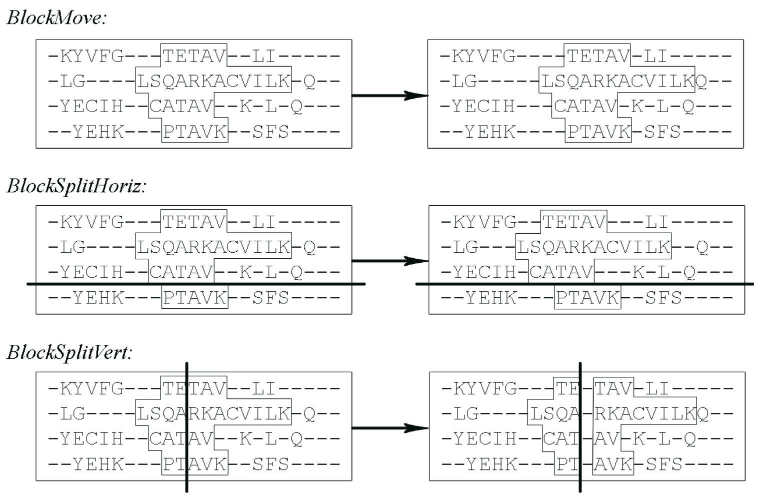

Fig. 2. *BlockShuffling operator* has the purpose to shifting blocks of amino acids or gaps. Upper plot shows the *BlockMove* operator; middle plot drawing how *Block-SplitHor* works, choosing the $4th$ row to divide the block in two level; lower plot shows *BlockSplitVer* operator performing a right shift.

Evaluate(P) computes the sum-of-pairs objective function of each B cell in the population P, i.e. the proposed alignment quality, using the equation 1.

The aging operator, used by the algorithm eliminates old B cells in the populations $P_d^{(t)}$, $P_{N_c}^{(gap)}$ and $P_{N_c}^{(block)}$, whilst maintaining high diversity in order to avoid premature convergence. The maximum number of generations a B cell can remain in the population is determined by the parameter τ_B :. When a B cell reaches $\tau_B + 1$ it is erased from the current population, even if it is a good candidate solution. The only exception is made for the best B cell present in the current population: (*Elitist-Aging*).

A new generation $P_d^{(t+1)}$ of d B cells is obtained by selecting the "survivors" after the aging operator was applied to the populations — the resulting populations are: ${}^aP_d^{(t)}$, ${}^aP_{N_c}^{(gap)}$ and ${}^aP_{N_c}^{(block)}$. The $(\mu + \lambda)$-selection operator (with $\mu = d$ and $\lambda = 2N_c$) reduces an offspring B cell population of size $\lambda \geq \mu$ to a new parent population of size μ. Such a selection operator guarantees monotonicity in the evolution dynamics.

Finally, T_{max} is the maximum number of fitness function evaluations and the termination criterion.

Table 1 shows the pseudo-code of the described hybrid immune algorithm. The functions Compute_Weights() and Normalize_Weights() compute and normalize the weights of the sequences using a rooted tree, which is used for the evaluation of the objective function.

4 The State-of-Art Methods for MSA

The most popular method to solve MSA is based on *Dynamic Programming* (DP) [9], which guarantees a mathematically optimal alignment. However, the method is limited to a small number of short sequences, since the size of the problem space increases with the number of sequences and their length. To overcome this problem several heuristic approaches, based on different strategies, have been developed to effectively deal with the computational complexity of the problem.

All current methodologies of multiple alignment are heuristics and can be classify in three main categories: *progressive alignments*, *exact algorithms* and *iterative alignments.*

Progressive Alignments. In progressive alignment methods multiple alignments are performed, first aligning the closest sequences and then the more distant ones are added. Although this approach has the advantage of being simplistic and very fast, it does not guarantee any level of optimization.

Therefore, the main drawback of this approach is that once a sequence has been aligned it cannot be modified, causing possible conflicts with successively added sequences. Alignment programs based on this approach are MULTALIGN [10], PILEUP [11], CLUSTALX [12], CLUSTALW [13], T-COFFEE [14]. Their strategy is to align sequences in a progressive manner using a consistency-based objective function in order to minimize possible errors. SPEM (sequence and Secondary-structure Profiles Enhanced Multiple alignment) [15] combines a sequence-based method with a consistency-based refinement for pairwise alignment, a progressive algorithm for multiple alignment and PROBCONS [16] a practical tool for progressive protein multiple sequence alignment based on *probabilistic consistency* which is a novel scoring function for measuring alignment quality.

Exact algorithms. In contrast to the previous approach, PIMA [17] uses local dynamic programming to align only the most conserved motifs. In the default setting it makes use of two alignment methods, maximum linkage and sequential branching, to decide the order of alignments ML_PIMA and SB_PIMA respectively. Exact algorithms were developed to align multiple sequences simultaneously [18]. High memory requirement, high computational effort and limitation on the number of sequences limit their usage. A new divide and conquer algorithm [19] extending their capabilities was developed.

Iterative alignments. Iterative alignment methods depend on algorithms able to produce an alignment and to refine it through a series of iterations until no further improvements can be made. They are based on the idea that the solution to a given problem can be computed by modifying an already existing *sub-optimal solution.* Aligners which are based on this approach are:

– DIALIGN [20,21], a consistency-based algorithm which attempts to use local information in order to guide a global alignment, i.e. to construct multiple alignments based on segment-to-segment comparisons — such segments are incorporated into a multiple alignment using an iterative procedure.

- PRRP [22] optimizes a progressive global alignment by iteratively dividing the sequences into two groups which are realigned using a global group-to-group alignment algorithm.
- HMMT [23] is based on Hidden Markov Model (HMM), using simulated annealing (SA) to maximize the probability that a HMM represents the sequences to be aligned.
- MUSCLE (multiple sequence comparison by log-expectation) [24] is based on similar strategies used by PRRP.
- SAGA (Sequence Alignment by Genetic Algorithm) [25] is a genetic algorithm based on COFFEE (Consistency Objective Function For alignmEnt Evaluation) objective function [26]. The model described in SAGA has received considerable interest in the evolutionary computation community.
- Another iterative alignment method is Praline [27]; it begins with a preprocessing of the sequence to align.

In general, Evolutionary Algorithms tend to be suitable tools for the MSA [28] and can be used to effectively search in large solution spaces. But they spend a lot of time gradually improving potential solutions before reaching a solution comparable to deterministic methodologies [29]. This is due to a random initialization of the candidate alignments.

5 Results

The immune algorithm presented has been tested on the classical benchmark BaliBASE version 1.0 and version 2.0. BAliBASE (Benchmark Alignment dataBASE) [36] is a database developed to evaluate and compare all multiple alignments programs containing high quality (manually refined) multiple sequence alignments.

BAliBASE is divided into two versions: the first version contains 141 reference alignments and is divided into five hierarchical reference sets containing twelve representative alignments. Moreover, for each alignment the *core blocks* are defined. They are the regions which can be reliably aligned and they represent 58% of residues in the alignments. The remaining 42% are in ambiguous regions which cannot be reliably aligned.

Reference 1 contains alignments of equi-distant sequences with similar length, reference 2 contains alignments of a family (closely related sequences with $> 25\%$ identity) and 3 "orphan" sequences with $< 20\%$ identity, reference 3 consists of up to four families with $< 25\%$ identity between any two sequences from different families and references 4 and 5 contain sequences with large N/C-terminal extensions or internal insertions. For an extensive explanation of all references please refer to [3].

In the second version, BAliBASE v.2.0 [37], all alignments present in the first version have been manually verified and it includes three new reference sets: repeats, circular permutations and transmembrane proteins. It consists of 167

Table 2. SP values given by several methods on the BAliBase v.1.0 benchmark (http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/) for multiple sequence alignment

Aligner	Ref 1(82)	Ref 2(23)	Ref 3(12)	Ref 4(12)	Ref 5(12)	Overall(141)
Hybrid CSA	80.7	**88.6**	**77.4**	70.2	82.0	**79.7**
DIALIGN [20]	77.7	38.4	28.8	**85.2**	**83.6**	62.7
CLUSTALX [12]	85.3	58.3	40.8	36.0	70.6	58.2
PILEUP8 [11]	82.2	42.8	33.3	59.1	63.8	56.2
ML_PIMA [17]	80.1	37.1	34.0	70.4	57.2	55.7
PRRP [22]	**86.6**	54.0	48.7	13.4	70.0	54.5
SAGA [25]	70.3	58.6	46.2	28.8	64.1	53.6
SB_PIMA [17]	81.1	37.9	24.4	72.6	50.7	53.3
MULTALIGN [10]	82.3	51.6	27.6	29.2	62.7	50.6

Table 3. Alignment accuracies given by several methods on the the BAliBASE v.2.0 benchmark (http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE2/) for multiple sequence alignment [15]

Aligner	Ref 1(82)		Ref 2(23)		Ref 3(12)		Ref 4(12)		Ref 5(12)		Overall(141)	
	SP	CS	SP	CS	SP	CS	SP	CS	SP	CS	SP	CS
SPEM [15]	**90.8**	83.9	93.4	57.3	81.4	56.9	**97.4**	**90.8**	97.4	**92.3**	**91.5**	78.6
MUSCLE [24]	90.3	**84.7**	64.4	60.9	82.2	61.9	91.8	74.8	**98.1**	92.1	91.0	**78.7**
ProbCons [16]	90.0	83.9	**94.0**	**62.6**	**82.3**	**63.1**	90.9	73.6	**98.1**	91.7	90.8	78.4
T-Coffee [14]	86.8	80.0	93.9	58.5	76.7	54.8	92.1	76.8	94.6	86.1	88.2	74.6
PRALINE [27]	90.4	83.9	**94.0**	61.0	76.4	55.8	79.9	53.9	81.8	68.6	88.2	73.9
ClustalW [13]	85.8	78.3	93.3	59.3	72.3	48.1	83.4	62.3	85.8	63.4	85.7	70.0
Hybrid CSA	82.7	65.3	91.9	41.3	78.6	36.2	70.5	31.9	83.6	56.9	81.4	46.3

reference alignments with more than 2100 sequences. The three new references contain 26 protein families with 12 distinct repeat types, 8 transmembrane families and 5 families with inverted domains.

Table 2 shows the average SP score obtained by the described alignment tools on every instance set of BAliBASE v.1.0. The values refer to the Sum of Pairs score, calculated by the "*baliscore.c*" program. As it can be seen in the table, Hybrid CSA performs well on the Ref 2 and Ref 3 sets. The values obtained aid to raise the overall score, which is higher compared to the results published by the Bioinformatic platform of Strasbourg[1].

Table 3 shows the average SP and Column Score (CS) values obtained by the compared tools on every group of instances belonging to the BAliBASE v.2.0 database. The Column Score is defined as the number of correctly aligned columns present in the generated alignments, divided by the total number of aligned columns in the core blocks of the reference alignment.

[1] http://bips.u-strasbg.fr/fr/Products/Databases/BAliBASE/

The values used in table3 are drawn from data reported in [15]. Hybrid CSA obtains comparable values of SP score on Ref 1, Ref2 and Ref 5 — despite the fact that the value obtained on Reference 3 is the fourth best value. This table also shows that future effort should focus on improving the CS metric.

6 Conclusions and Future Works

Experimental results of benchmark BaliBASE v.1.0 show that the proposed algorithm is superior to PRRP, ClustalX, SAGA, DIALIGN, PIMA, MULTIALIGN and PILEUP8. Whilst on BaliBASE v.2.0 the algorithm shows interesting results in terms of SP score with respect to established and leading methods, e.g. ClustalW, T-Coffee, MUSCLE, PRALINE, ProbCons and Spem.

A strong point of the IA is the ability of generating more than a single alignment for every MSA instance. This behaviour is due to the stochastic nature of the algorithm and the populations evolved during the convergence process. Another advantage of the aligner is that the alignment process is not affected by the presence of distant sequences in the starting protein set. As shown by the experimental results, the scoring function used by the IA produces high SP values and low CS scores, therefore future work will first focus on the improvement of the CS score values using the T-Coffee scoring function. The second step will be the more accurate tuning of the parameters and the operators in order to improve the convergence speed.

Acknowledgments

This work was supported by the National Research Laboratory Grant (2005-01450) from the Ministry of Science and Technology. D.L. and M.P. would like to thank CHUNG Moon Soul Center for BioInformation and BioElectronics for providing research and computing facilities.

References

1. Eidhammer I., Jonassen I., Taylor W. R.; "*Protein Bioinformatics*," Chichester, West Sussex, UK, Wiley, (2004)
2. Durbin R., Eddy S., Krogh A., Mitchison G.; "*Biological sequence analysis*", Cambridge, UK, Cambridge University Press (2004)
3. Thompson J. D., Plewniak F., Ripp R., Thierry J.C., Poch O.; "Towards a Reliable Objective Function for Multiple Sequence Alignments", in J. Mol. Biol., vol. 301, pp. 937-951 (2001)
4. Altschul S. F., Lipman D. J.; "*Trees stars and multiple biological sequence alignment*," in SIAM Journal on Applied Mathematics, vol. 49, pp. 197–209, (1989).
5. Altschul S. F., Carroll R. J., Lipman D. J.; "*Weights for data related by a tree*," in Journal on Molecular Biology, vol. 207, pp. 647–653, (1989).
6. Bonizzoni P., Della Vedova G.; "*The Complexity of Multiple Sequence Alignment with SP-score that is a Metric*," in Theoretical Computer Science, vol. 259 (1), pp. 63–79 (2001).

7. Wang L., Jiang T.; "*On the complexity of multiple sequence alignment*," in Journal of Computational Biology, vol. 1, pp. 337–348, (1994)
8. Garey M. R., Johnson D. S.; "*Computers and Intractability: A Guide to the Theory of NP-Completeness*," Freeman, New York (1979).
9. Gupta S. K., Kececioglu, J. D., Schaffer A.; "*Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment*," in Journal of Computational Biology, vol. 2, pp. 459–472, (1995)
10. Corpet F.; "*Multiple sequence alignment with hierarchical clustering*," in Nucleic Acids Research, vol. 16, pp. 10881–10890, (1988)
11. Wisconsin Package v.8; Genetics Computer Group, Madison, WI. www.gcg.com
12. Thompson J. D., Gibson T. J., Plewniak F., Jeanmougin F., Higgins D. G.; "*The ClustalX windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools*," in Nucleic Acids Research, vol. 24, pp. 4876–4882, (1997)
13. Thompson J. D., Higgins D. G., Gibson T. J.; "*CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice*," in Nucleic Acids Research, vol. 22, pp. 4673–4680, (1994)
14. Notredame C., Higgins D. G., Heringa J.; "*T-Coffee: a novel method for fast and accurate Multiple Sequence Alignment*," in Journal Molecular Biology, vol. 302, pp. 205–217, (2000)
15. Zhou H., Zhou Y.; "*SPEM: Improving multiple sequence alignment with sequence profiles and predicted secondary structures*," in Bioinformatics, vol. 21, pp. 3615–3621, (2005)
16. Do C. B., Mahabhashyam M. S. P., Brudno M., Batzoglou S.; "*ProbCons: Probabilistic consistency-based multiple sequence alignment*," in Genome Research, vol. 15, pp. 330–340, (2005)
17. Smith R. F., Smith T. F.; "*Pattern-induced multi-sequence alignment (PIMA) algorithm employing secondary structure-dependent gap penalties for use in comparative protein modelling*," in Protein Engineering, vol. 5, pp. 35–41,(1992).
18. Carrillo H., Lipman D. J.; "*The Multiple Sequence Alignment Problem in Biology*," in SIAM Journal on Applied Mathematics, vol. 48, pp. 1073–1082, (1988)
19. Stoye J., Moulton V., Dress A. W.; "*DCA: an efficient implementation of the divide-and conquer approach to simultaneous multiple sequence alignment*," in Bioinformatics, vol. 13 (6), pp. 625–626, (1997)
20. Morgenstern B., Frech K., Dress A., Werner T.; "*DIALIGN: Finding local similarities by multiple sequence alignment*," in Bioinformatics, vol. 14, pp. 290–294,(1998)
21. Morgenstern B., Frech K., Dress A., Werner T.; "*DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment*," in Bioinformatics, vol. 15, pp. 211–218, (1999)
22. Gotoh O; "*Further improvement in methods of group-to-group sequence alignment with generalized profile operations*," in Bioinformatics, vol. 10 (4), pp. 379–387, (1994)
23. Eddy S. R.; "*Multiple alignment using hidden Markov models*," in 3*rd* International Conference on Intelligent Systems for Molecular Biology, vol. 3, pp. 114–120, (1995)
24. Edgar R. C.; "*MUSCLE: multiple sequence alignment with high accuracy and high throughput*," in Nucleic Acids Research, vol. 32, pp. 1792–1797, (2004)
25. Notredame C., Higgins D.G.; "*SAGA: sequence alignment by genetic algorithm*," in Nucleic Acids Research, vol. 24, pp. 1515–1539, (1996)
26. Notredame C.; "*COFFEE: an objective function for multiple sequence alignments*," in Bioinformatics, vol. 14, pp. 407–422, (1998)

27. Simossis V. A., Heringa J.; "*PRALINE: a multiple sequence alignment toolbox that integrates homology-extended and secondary structure information*," in Nucleic Acids Research, vol. 33, pp. 289–294, (2005)
28. Shyu C., Sheneman L., Foster J. A.; "*Multiple Sequence Alignment with Evolutionary Computation*," in Genetic Programming and Evolvable Machines, vol. 5, pp. 121-144, (2004)
29. Nguyen H. D., Yoshihara I., Yamamori K., Yasunaga M.; "*Aligning Multiple Protein Sequences by Parallel Hybrid Geneti Algorithm*," in Genome Informatics, vol. 13, pp. 123–132, (2002)
30. Cutello V., Nicosia G.; "*An Immunological Approach to Combinatorial Optimization Problems*", Advances in Artificial Intelligence - IBERAMIA 2002, 8th Ibero-American Conference on AI, Seville, Spain, November 12-15, 2002. Springer, Lecture Notes in Computer Science, vol. 2527, pp. 361-370, (2002)
31. Nicosia G.; "*Immune Algorithms for Optimization and Protein Structure Prediction*," Ph.D. Dissertation, Department of Mathematics and Computer Science, University of Catania, Italy (2004)
32. Cutello V., Narzisi G., Nicosia G., Pavone M.; "*Clonal selection algorithms: A comparative case study using effective mutation potentials*," in 4th International Conference on Artificial Immune Systems (ICARIS), pp. 13–28 (2005)
33. Cutello V., Nicosia G., Pavone M., Timmis J.; "*An Immune Algorithm for Protein Structure Prediction on Lattice Models*," to appear in IEEE Transaction on Evolutionary Computation.
34. Cutello V., Nicosia G., Pavone M.; "*Exploring the capability of immune algorithms: A characterization of hypermutation operators*," in 3rd International Conference on Artificial Immune Systems (ICARIS), pp. 263–276 (2004)
35. Taylor W. R.; "*A flexible method to align a large number of sequences*," in J. Mol. Evol., vol. 28, pp. 161-169, (1988)
36. Thompson J. D., Plewniak F., Poch O.; "*BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs*," in Bioinformatics, vol. 15, pp. 87–88 (1999).
37. Bahr A., Thompson J. D., Thierry J. C., Poch O.; "*BAliBASE (Benchmark Alignment dataBASE): Enhancements for Repeats, Transmembrane Sequences and Circular Permuations*," in Nucleic Acids Research, vol. 29 (1), pp. 232–326 (2001).