

# ANTICLUSTAL: Multiple Sequence Alignment by Antipole Clustering and Linear Approximate 1-Median Computation

C. Di Pietro<sup>3</sup>, V. Di Pietro<sup>3</sup>, G. Emmanuele<sup>3</sup>, A. Ferro<sup>1</sup>, T. Maugeri<sup>1</sup>,  
E. Modica<sup>3</sup>, G. Pigola<sup>1</sup>, A. Pulvirenti<sup>1</sup>, M. Purrello<sup>3</sup>, M. Ragusa<sup>3</sup>,  
M. Scalia<sup>3</sup>, D. Shasha<sup>2</sup>, S. Travali<sup>3</sup>, V. Zimmitti<sup>3</sup>.

<sup>1</sup> *Dipartimento di Matematica e Informatica Università di Catania*

e-mail: {ferro,tarcisio,pigola,apulvirenti}@dmi.unict.it

<http://alpha.dmi.unict.it/~ctnyu/>

<sup>2</sup> *Courant Institute of Mathematical Science New York University*

e-mail: shasha@cs.nyu.edu

<sup>3</sup> *Dipartimento di Scienze Biomediche, Università di Catania*

e-mail: purrello@unict.it

## Abstract

*In this paper we present a new Multiple Sequence Alignment (MSA) algorithm called AntiClustAl. The method makes use of the commonly used idea of aligning homologous sequences belonging to classes generated by some clustering algorithm, and then continue the alignment process in a bottom-up way along a suitable tree structure. The final result is then read at the root of the tree. Multiple sequence alignment in each cluster makes use of the progressive alignment with the 1-median (center) of the cluster. The 1-median of set  $S$  of sequences is the element of  $S$  which minimizes the average distance from any other sequence in  $S$ . Its exact computation requires quadratic time. The basic idea of our proposed algorithm is to make use of a simple and natural algorithmic technique based on randomized tournaments which has been successfully applied to large size search problems in general metric spaces. In particular a clustering algorithm called Antipole tree and an approximate linear 1-median computation are used. Our algorithm compared with Clustal W, a widely used tool to MSA, shows a better running time results with fully comparable alignment quality. A successful biological application showing high aminoacid conservation during evolution of *Xenopus laevis* SOD2 is also cited.*

**Keywords:** *Multiple Sequence Alignment, Clustering, Randomized Tournament.*

## 1 Introduction

Multiple sequence alignment is the process of taking three or more input sequences and forcing them to have the same length by inserting a universal gap symbol - in order to maximize their similarity as measured by a score function. In the case of biological sequences (DNA, RNA, Protein) the resulting aligned sequences can be used for two purposes: first, to find regions of similarity defining a conserved consensus pattern of characters (nucleotides or amino acids) in all of the sequences; second, if the alignment is particularly strong, to use the aligned positions to derive the possible evolutionary relationships among the sequences. A group of similar sequences may define a protein family which may share a common biochemical function or evolutionary origin. More formally let  $\Sigma$  be an alphabet and  $S = \{S_1, \dots, S_k\}$  be a set of string defined over  $\Sigma$ . A *multiple sequence alignment* of  $S$  is a set  $S' = \{S'_1, \dots, S'_k\}$  such that:

- $S'_i \in (\Sigma \cup \{-\})^*$  for each  $i = 1, \dots, k$ ;
- $S_i$  is obtained from  $S'_i$  by dropping all gap symbols  $\{-\}$ ;
- $|S'_1| = |S'_2| = \dots = |S'_k|$ .

A *scoring function* defined on the alphabet  $\Sigma$  is a map  $\sigma : (\Sigma \cup \{-\})^k \mapsto R$ . It has the following properties:

1. Reflexivity (maximum score if all the same)  
 $\sigma(a, \dots, a) \geq \sigma(a_1, \dots, a_k)$ , provided  $a \neq -$ .
2. Symmetry (it doesn't matter where differences are found):

$$\begin{aligned} & \sigma(x_1, \dots, x_i, a, x_{i+2}, \dots, x_j, b, x_{j+2}, \dots, x_k) \\ & = \sigma(x_1, \dots, x_i, b, x_{i+2}, \dots, x_j, a, x_{j+2}, \dots, x_k) \end{aligned}$$

3. Triangle inequality (recall that similarity is the opposite of distance):

$$\begin{aligned} & \sigma(x_1, \dots, x_i, a, x_{i+2}, \dots, x_j, b, x_{j+2}, \dots, x_k) \\ & + \sigma(x_1, \dots, x_i, b, x_{i+2}, \dots, x_j, c, x_{j+2}, \dots, x_k) \geq \\ & \geq \sigma(x_1, \dots, x_i, a, x_{i+2}, \dots, x_j, c, x_{j+2}, \dots, x_k) \end{aligned}$$

The best score  $D(|S_1|, |S_2|, \dots, |S_k|)$  for aligning  $k$  sequences  $S_1, S_2, \dots, S_k$  with respect to  $\sigma$  is the one that maximizes the sum of the  $\sigma$ s across all positions:  $\sum_{i \in 1..k} \sigma(S_1[i], S_2[i] \dots S_k[i])$ . If  $|S_1| = |S_2| = |S_k| = n$  then the space and the time complexity of the best currently known algorithm is  $\mathcal{O}(n^k)$  and  $\mathcal{O}(2^k n^k) \times \mathcal{O}(\text{computation of the } \sigma \text{ function})$  respectively. Finding the optimal solution of the multiple sequence alignment therefore requires exponential space and time complexity. If only pairwise alignment is considered then an  $\mathcal{O}(n^2 / \log n)$  algorithm can be obtained [6].

The most successful solution to the above problem has been provided by the program Clustal W [11]. In this paper we propose a new solution based on a top down “bisector tree” [5] clustering algorithm called Antipole Tree in connection with a linear approximate 1-median computation. Since exact 1-median computation requires a quadratic number of distance calculations and given that each of such distance computations may require quadratic time in the length of the biosequences, then the use of a linear approximate 1-median computation may give a much better running time. Both clustering and approximate 1-median computation algorithms make use of a very simple and natural technique based on randomized tournaments. Given a set  $S$  of sequences to be aligned we play the following tournament. At each round, the winners of the previous round are randomly partitioned into subsets of a fixed size  $t$ . Then a procedure finds each subset’s 1-median and discards it. Rounds are played until less than  $2t$  elements are left. The farthest pair of points in that set is our antipole pair  $S_1, S_2$  of elements.

If the distance (pairwise alignment) of our antipole pair lies below a given threshold then splitting is not executed and a new cluster is generated. Otherwise partition the input set of elements according to their proximity to  $S_1, S_2$ . Each resulting class is then treated as a new input set recursively. The process terminates with a given set of clusters stored in the leaves of the generated Antipole tree. A similar randomized tournament process can be applied to each cluster to generate is approximate 1-median. Let  $C$  be such a cluster, at each round, the winners of the

previous round are randomly partitioned into subsets of a fixed size  $t$  and a local optimization procedure is used to determine the winner for each subset which is the 1-median of the subset. Rounds are played until only one element, the *final winner*, is left.

The paper is organized as follows, section 2 is a short survey of preceding work. Section 3 introduces the Antipole tree for generic metric spaces. Section 4 presents the AntiClustAl Alignment Algorithm, Section 5 shows experimental results and comparison with Clustal W. Section 6 mentions a successful biological application whose details can be found in [15]. Conclusions and future development are given in section 7.

## 2 Related Work

In order to reduce the complexity of multiple sequence alignment several approaches have been proposed. The *Carillo Lipmann method* [4] provides a heuristic to accelerate the speed of the alignment process. The exponential time is lowered by a constant factor, since, if the sequences are very similar, the optimal solution can be discovered by visiting a hypercube in a small neighborhood of the main diagonal of the exponential dynamic programming algorithm.

Several approximation algorithms have been proposed. Two of them are particularly interesting. The first one [7, 10] uses a progressive alignment strategy that consists of incrementally aligning every sequence  $S_i \in S - \{S_c\}$  to the centroid  $S_c$  of  $S$ . The second, called *Clustal W* [11, 12], a widely used tool for solving the multiple sequence alignment problem for biosequences, starts by building a phylogenetic (evolutionary) tree [8]. *Clustal W* consists of three main stages. First of all, the pairwise distance matrix is computed on all the pairs of sequences. Next, the algorithm uses the *Neighbor-Joining Method* [16] to construct a phylogenetic tree. Finally all the sequences are aligned progressively from the leaves up.

At the leaf level, the process begins by pairwise alignment of two single strings and it proceeds by pairwise alignment of two sets of strings which are the result of previous alignments of children nodes. This last stage will make use of the concept of profile [9].

Other relevant works include [13, 1].

Let  $M$  be a multiple sequence alignment of length  $l$  defined over the alphabet  $\Sigma$ . A *profile*  $P$  is a  $l \times |\Sigma \cup \{-\}|$  matrix, whose columns are probability vectors denoting the frequencies of each symbol in the corresponding alignment column. The Profile concept is used in ClustalW as a mathematical instrument to perform either the alignment of a sequence  $S$  with a multiple sequence alignment  $M$  or the alignment of

two multiple alignments  $M_1$  and  $M_2$ . To align a profile  $P = (p_{i,j})$  for  $i = 1, \dots, l$  and  $j = 1, \dots, |\Sigma| + 1$  against a sequence  $S = s_1 s_2 \dots s_n$  one can use the classical algorithm by Miller and Myers [14] appropriately modified. Let  $\sigma : (\Sigma \cup \{-\}) \times (\Sigma \cup \{-\}) \mapsto \mathbb{R}$  be the scoring function defined by the rule:

$$\sigma(a, b) = \begin{cases} x & a = b \\ y & \text{otherwise} \end{cases}$$

where  $x, y$  are two different real numbers. Let  $\hat{\sigma} : (\Sigma \cup \{-\}) \times \{1, 2, \dots, l\} \mapsto \mathbb{R}$  be a new weighted average scoring function defined as follows:

$$\hat{\sigma}(b, i) = \sum_{a \in \Sigma} p_{i,a} \cdot \sigma(a, b) \quad (1)$$

where  $p_{i,a}$  represents the frequency of the base  $a$  in the  $i^{\text{th}}$  column of the profile  $P$ . By replacing the scoring function  $\sigma$  by  $\hat{\sigma}$  the Miller and Myers algorithm [14] reduces the alignment-to-sequence comparison problem to a sequence-to-sequence one.

Profiles are used to align two multiple sequence alignments  $M_1$  and  $M_2$  whose profiles are respectively  $P_1 = (p'_{i,k})$  and  $P_2 = (p''_{j,k})$  with  $i = 1, \dots, l$ ,  $j = 1, \dots, m$  and,  $k = 1, \dots, |\Sigma| + 1$ . Now the new scoring function  $\tilde{\sigma} : \{1, \dots, l\} \times \{1, \dots, m\} \mapsto \mathbb{R}$  takes as input two positions  $i$  in profile  $P_1$  and  $j$  in the profile  $P_2$  and returns the following value:

$$\tilde{\sigma}(i, j) = \sum_{k=1}^{|\Sigma|+1} f(p'_{i,k} \cdot p''_{j,k}) \quad (2)$$

where  $f(\cdot)$  is any monotonic function. By replacing the scoring function  $\sigma$  by  $\tilde{\sigma}$ , ClustalW reduces the problem of comparing a profile to a profile or a profile to a sequence to the problem of aligning two sequences  $M_1$  and  $M_2$ .

### 3 The Antipole Tree Data Structure for Clustering in Metric Spaces.

Let  $X$  be a finite set of objects (for example biosequences) and let  $d$  a distance function  $dist : X \times X \mapsto \mathbb{R}$  such that the following four properties hold:

1.  $dist(x, y) \geq 0 \forall x, y \in X$  (positiveness);
2.  $dist(x, y) = dist(y, x) \forall x, y \in X$  (symmetry);
3.  $dist(x, x) = 0 \forall x \in X$  (reflexivity);
4.  $dist(x, y) \leq dist(x, z) + dist(z, y) \forall x, y, z \in X$  (triangularity);

Clustering  $X$  with a bounded diameter  $\sigma$  is the problem of partitioning  $X$  in non empty subsets (clusters) of diameter less than  $\sigma$ . A centroid or the 1-median of a cluster  $Clu$  is the element  $C$  of  $Clu$  which minimize the following value  $\sum_{y \in X} d(C, y)$  and the radius of a cluster  $Clu$  is the distance between the centroid  $C$  and its furthest object in that cluster. Assume we fix a cluster diameter  $\sigma$  such that sequences whose pairwise distance is greater than  $\sigma$  are dissimilar enough. The Antipole clustering of bounded diameter  $\sigma$  [3] is performed by a top down procedure starting from a given finite set of points (biosequences in our case)  $S$  by a splitting procedure (see Fig. 1 (a)) which assigns each point of the splitting subset to the closest endpoint of a "Pseudo-Diameter" segment called *antipole*.<sup>1</sup> An Antipole pair of elements is generated as the final winner of a set of randomized tournaments (see Fig. 2). The initial tournament is made up by randomly partition the set  $S$  in  $t$ -uples (subsets of cardinality  $t$ ) and playing the following game in each  $t$ -uple. The winning subset of each  $t$ -uple is then obtained by discarding from it the local 1-median (see Fig. 2 (a)). The winning sets go to the next tournament whereas losing points are thrown away. If any of the tournaments has cardinality which is not a multiple of  $t$  then one of the games will have at most  $2t - 1$  players. If the final winner pair has a distance (pairwise alignment) which is lower than  $\sigma$  then splitting is not performed and the subset is one of the clusters (see Fig. 1 (b)). Otherwise splitting is performed and one proceeds recursively on each of the two generated subsets. At the end of the procedure the centroid of each cluster [2] is computed by an algorithm similar to the one to find the antipole pair (see Fig. 2 (b)). In this case the winner of each game is the object that minimize the sum from the other two. This procedure gives rise to a data structure called Antipole Tree.

### 4 AntiClustAl: Multiple Sequence Alignment via Antipoles

In this section we show that replacing the phylogenetic tree with the antipole tree gives a substantial speed improvement to the ClustalW approach with as good or better quality. (The quality of our approach derives from the fact that multiple alignment of a set of sequences works better when the diameter of the set of sequences is small.) Our basic algorithm is: build the antipole tree. Then align the sequences progressively from the leaves up, as done in ClustalW. Here are the details of the last stage.

<sup>1</sup>The Pseudo-Diameter in such a case is a pair of biosequences, the endpoints, different enough.

<pre> BUILD_TREE(<math>S, \sigma</math>) 1  <math>Q \leftarrow \text{APPROX\_ANTIPOLE}(S, \sigma)</math>; 2  <b>if</b> <math>Q = \emptyset</math> <b>then</b> // splitting condition fails 3    <math>T.\text{Leaf} \leftarrow \text{TRUE}</math>; 4    <math>T.C \leftarrow \text{MAKE\_CLUSTER}(S)</math>; 5    <b>return</b> <math>T</math>; 6  <b>end if</b>; 7  <math>\{A, B\} = Q</math>; // A, B are the antipoles sequences 8  <math>T.A \leftarrow A</math>; 9  <math>T.B \leftarrow B</math>; 10 <math>S_A \leftarrow \{O \in S \mid \text{dist}(O, A) &lt; \text{dist}(O, B)\}</math>; 11 <math>S_B \leftarrow \{O \in S \mid \text{dist}(O, B) \leq \text{dist}(O, A)\}</math>; 12 <math>T.\text{left} \leftarrow \text{BUILD\_TREE}(S_A, \sigma)</math>; 13 <math>T.\text{right} \leftarrow \text{BUILD\_TREE}(S_B, \sigma)</math>; 14 <b>return</b> <math>T</math>; 15 <b>END BUILD\_TREE.</b> </pre> <p style="text-align: center;">(a)</p>	<pre> MAKE_CLUSTER(<math>S</math>) 1  <math>C.\text{Centroid} \leftarrow \text{APPROX\_1\_MEDIAN}(S)</math>; 2  <math>C.\text{Radius} \leftarrow \max_{x \in S} \text{dist}(x, C.\text{Centroid})</math>; 3  <math>C.C_{\text{List}} \leftarrow S \setminus \{C.\text{Centroid}\}</math>; 4  <b>return</b> <math>C</math>; <b>END MAKE\_CLUSTER.</b> </pre> <p style="text-align: center;">(b)</p>
--	--

Figure 1: (a) Antipole Algorithm, (b) MakeCluster Algorithm.

<p style="text-align: center;"><b>The approximate Antipole selection algorithm</b></p> <pre> LOCAL_WINNER(<math>T</math>) 1  <b>return</b> <math>T \setminus \text{1-MEDIAN}(T)</math>; 2  <b>END LOCAL\_WINNER</b>  FIND_ANTIPOLE(<math>T</math>) 1  <b>return</b> <math>P_1, P_2 \in T</math> such that     <math>\text{dist}(P_1, P_2) \geq \text{dist}(x, y) \forall x, y \in T</math>; 2  <b>END FIND\_ANTIPOLE</b>  APPROX_ANTIPOLE(<math>S</math>) 1  <b>while</b> <math> S  &gt; \text{threshold}</math> <b>do</b> 2    <math>W \leftarrow \emptyset</math>; 3    <b>while</b> <math> S  \geq 2 * t</math> <b>do</b> 4      <i>Randomly choose a set</i> <math>T \subseteq S :  T  = t</math>; 5      <math>S \leftarrow S \setminus T</math>; 6      <math>W \leftarrow W \cup \{\text{LOCAL\_WINNER}(T)\}</math>; 7    <b>end while</b> 8    <math>S \leftarrow W \cup \{\text{LOCAL\_WINNER}(S)\}</math>; 9  <b>end while</b> 10 <b>return</b> <math>\text{FIND\_ANTIPOLE}(S)</math>; 11 <b>END APPROX\_ANTIPOLE</b> </pre> <p style="text-align: center;">(a)</p>	<p style="text-align: center;"><b>The approximate 1-Median selection algorithm</b></p> <pre> 1-MEDIAN(<math>X</math>) 1  <b>for each</b> <math>x \in X</math> <b>do</b> 2    <math>\sigma_x \leftarrow \sum_{y \in X} d(x, y)</math>; 3    Let <math>m \in X</math> be an element such that     <math>\sigma_m = \min_{x \in X} (\sigma_x)</math>; 4  <b>return</b> <math>m</math> 5  <b>END 1-MEDIAN</b>  APPROX_1_MEDIAN(<math>S</math>) 1  <b>while</b> <math> S  &gt; \text{Threshold}</math> <b>do</b> 2    <math>W \leftarrow \emptyset</math>; 3    <b>while</b> <math> S  \geq 2t</math> <b>do</b> 4      <i>Randomly choose a set</i> <math>T \subseteq S :  T  = t</math>; 5      <math>S \leftarrow S \setminus T</math>; 6      <math>W \leftarrow W \cup \{\text{1-MEDIAN}(T)\}</math>; 7    <b>end while</b>; 8    <math>S \leftarrow W \cup \{\text{1-MEDIAN}(S)\}</math>; 9  <b>end while</b>; 10 <b>return</b> <math>\text{1-MEDIAN}(S)</math>; 11 <b>END APPROX\_1\_MEDIAN</b> </pre> <p style="text-align: center;">(b)</p>
---	--

Figure 2: Pseudo-code of the approximate algorithms: (a) Approximate Antipole search, (b) 1-median Computation. The variable *threshold* is usually taken to be  $(t^2 + 1)$ . Indeed this is the lowest value for which it is possible to partition the set  $S$  in subsets of size between  $t$  and  $2t - 1$ . The subset size  $t$  is normally taken equal to three. This guarantees good performance. However it can be experimentally shown that the optimal choice of  $t$  is equal to the dimension of the underlying metric space plus one.

The final stage of multiple sequence alignment begins from the leaves, aligning all the sequences of the corresponding cluster using the profile alignment technique. Figs. 3,4 and, 5 contain the pseudo code of the multiple sequence alignment via antipole. The recursive function *AntiClustal* visits the Antipole tree from the leaves up (just as ClustalW visits the phylogenetic tree). It aligns all the sequences stored in the leaves (the clusters) by calling the function *AlignCluster*. Next two aligned clusters are merged by the function *MergeAlignment*. The three mentioned procedures make use of the functions *AlignSequences*, *AlignProfileVsSequence* and, *OptimalAlignment* which align respectively two profiles, a profile vs a sequence, and two sequences according to the Miller and Myers algorithm [14]. Finally the function *GetProfile* returns the profile of a multiple sequence alignment. Fig. 6 shows an example of the proposed method.

```

ANTICLUSTAL(TREE T)
1  if (!isLeaf(T)) /* true if T is a leaf */
2    AntiClustal(T.left);
3    AntiClustal(T.right);
4    T ← MergeAlignment(T.left, T.right);
5    T.leaf ← TRUE;
6    return(T);
7  else
8    AlignCluster(T);
9    return (T);
10 end

```

Figure 3: Multiple Sequence Alignment via the Antipole Tree.

```

MERGEALIGNMENT(TREE A, TREE B)
1  A ← AlignCluster(A);
2  B ← AlignCluster(B);
3  P1 ← GetProfile(A);
4  P2 ← GetProfile(B);
5  C ← AlignSequences(A, P1, B, P2);
6  return C.

```

Figure 4: How to align twoclusters

## 5 Comparing ClustalW and Anti-ClustAl

In this section we present some experiments to compare ClustalW1.82 with *Antipole Clustering Alignment* AntiClustal. The comparison has been made in terms of both running time and precision. The experiments were performed on alpha-globins, beta-globins and immuno-globulins downloaded from *GenBank*. Each dataset had size from 10 to 200 biosequences. Each biosequence had length from 120 to 100000

```

ALIGNCLUSTER(TREE A)
1  if (|A.cluster| = 1)
2    return A;
3  else
4    C ← OptimalAlignment(A0, A1);
5    if (|A| ≥ 3)
6      for each Ai ∈ A.cluster do
7        P ← GetProfile(C);
8        C ← AlignProfileVsSequence(Ai, P, C);
9      end for
10     end if
11     return C;
12 end;

```

Figure 5: How to align the sequences in a cluster.

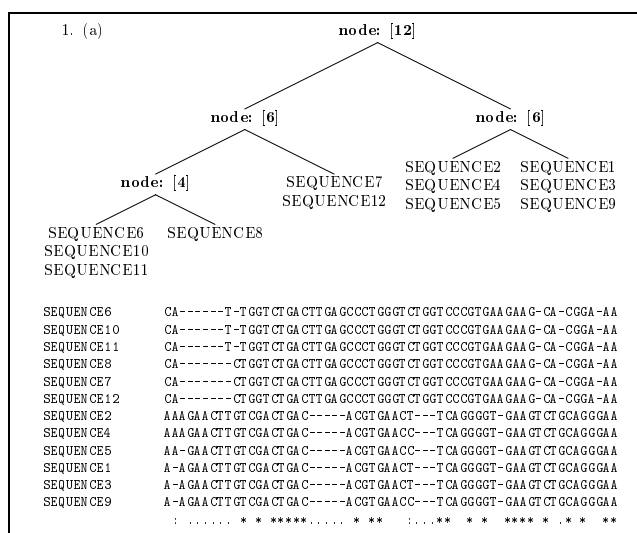


Figure 6: An example of multiple sequence alignment via Antipole Tree using alpha-globin sequences. The picture in the bottom side shows a portion of the aligned sequences. The symbols under the aligned sequences show the kind of the matches concord the given definition.

bases. Precision was measured by the relative match number which is obtained by dividing the match number by the alignment length. Specifically, we put:

$$MatchRatio = 100 \cdot \frac{match\ number}{alignment\ length}$$

A biologically more significant comparison parameter is the *column blocks match*. This is obtained by associating an integer to each column of a multiple alignment. Call that integer *column match* and set its value to between 0 and 3 according to the following scheme.

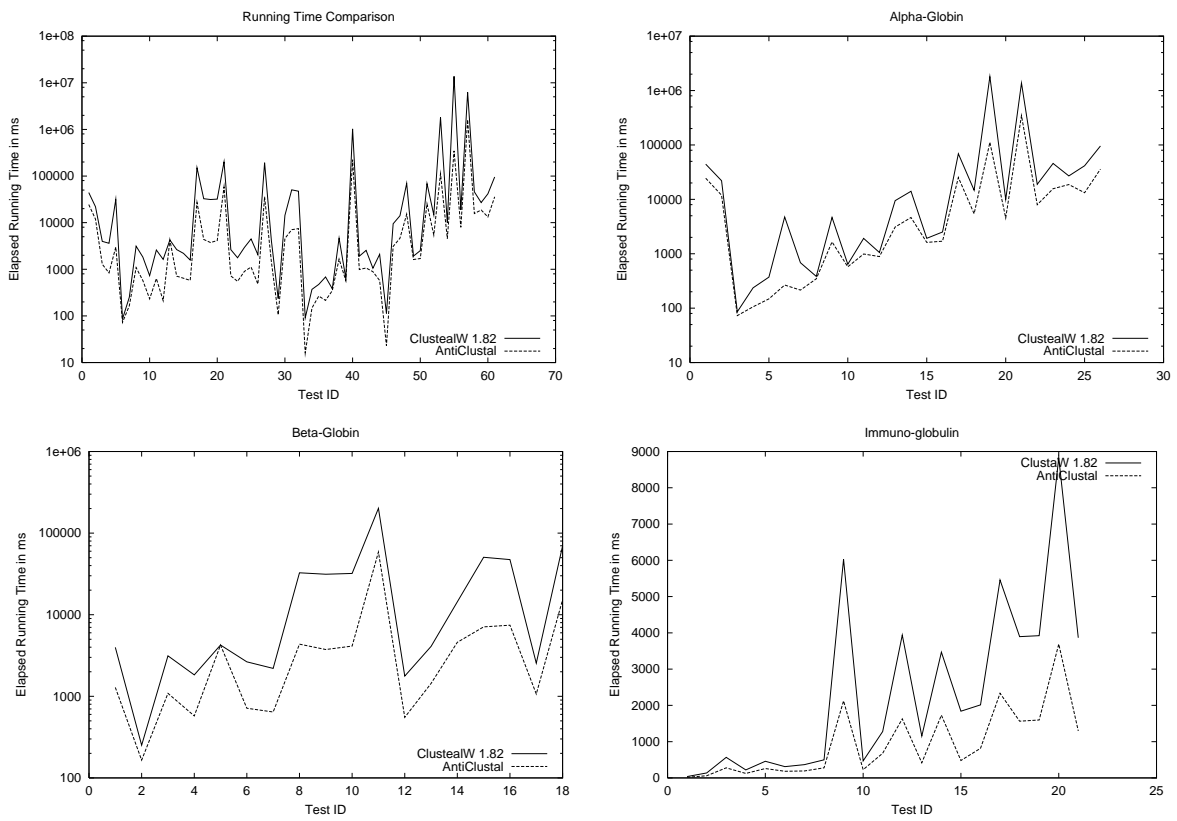


Figure 7: Running time Comparison between ClustalW1.82 and AntiClustal

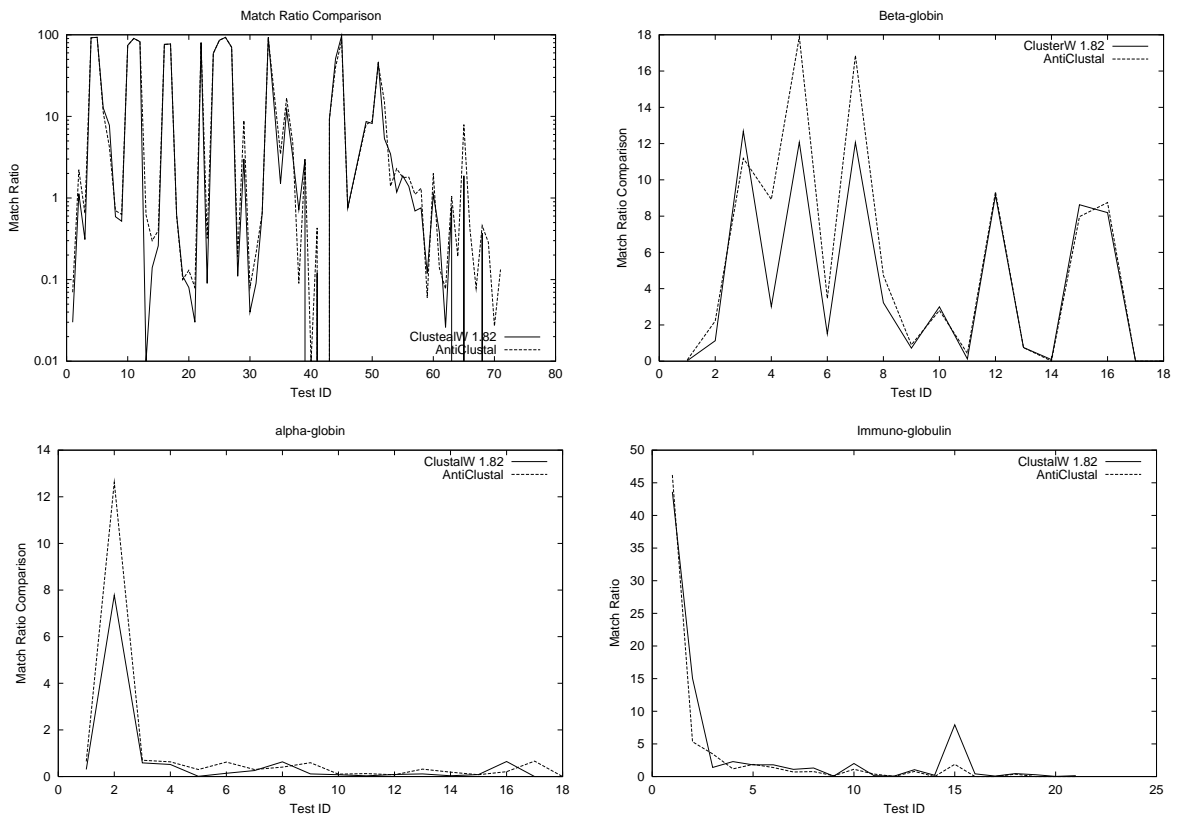


Figure 8: Match Comparison between ClustalW1.82 and AntiClustal

- 3 if the  $i - th$  column contains a *perfect match*, that is every element in the  $i^{th}$  column is the same non-gap symbol;
- 2 if the  $i - th$  column contains a *weak match*, that is, 75% of its elements are the same non-gap symbol;
- 1 if the  $i - th$  column contains a *trivial match*, that is, more than 50% of its elements are the same non-gap symbol and the remaining elements are all *gap* symbols;
- 0 otherwise. In this case we have a *mismatch*.

Fig. 7 shows that the running time of the Antipole clustering alignment is better than ClustalW because the Antipole algorithm performs a linear rather than quadratic number of distance computations because AntiClustal constructs an antipole tree rather than a phylogenetic tree. (Thus the advantage increases as the number of sequences increases.) Concerning the quality of the alignment, Fig. 8, 9, and 10 show that antipole clustering alignment usually gives a higher quality result than ClustalW in terms of both *match ratio* and *column blocks match*.

## 6 Biological applications

The determination of the nucleotide sequence of a cDNA (complementary DNA) molecule and, indirectly, of the aminoacid sequence of the encoded polypeptide, as well as their comparison with other sequences available in databases, are essential steps in the analysis of the physiological functions of a protein and its potential involvement in pathology. This simple statement well explains why Computational Biology (otherwise called Bioinformatics) has become so important in contemporary Biomedical Research. Superoxyde Dismutases (SODs) are ubiquitary enzymes present in all organisms (both Prokaryotes and Eukaryotes) using oxygen for their energy metabolism: accordingly, they are very ancient molecules. Four different types of SOD have been identified to date: Cu / Zn SOD (SOD1), Mn SOD (SOD2), Fe SOD, Ni SOD. Eukaryotic SOD2 is encoded by a nuclear gene but is localized within the mitochondrion, where it catalyzes the dismutation of  $O_2^-$  and possibly of other ROS (Reactive Oxygen Species) into  $H_2O_2$ . ROS are highly reactive and obligatory byproducts of chemical reactions involving molecular oxygen, that are produced mainly within the mitochondrion: by blocking their harmful effects on essential cell molecules (such as nucleic acids, proteins and phospholipids) at their major site of production, SOD2 performs a critical

role in metabolism since it allows cells to use oxygen as final acceptor of electrons, derived from food stuff and carried by NADH and  $FADH_2$ : through this pathway, organisms obtain a much greater amount of energy than that allowed by glycolysis alone, without the high price of heavy chemical damage to be otherwise payed. By using the methodology of both wet and dry biology (i.e., RT-PCR, cycle sequencing and biocomputational technology, respectively) and the data obtained through the Genome Projects, we have cloned and determined the primary structure of the full length coding region of *Xenopus laevis* SOD2 (MnSOD2), which was chosen for our studies based on the peculiar evolutionary position of Amphibians [15]. These data, together with the deduced aminoacid sequence of the protein, were then compared with all the other SOD2 nucleotide and aminoacid sequences from eukaryotes and prokaryotes, present in the public databases. The analysis was performed by using both Clustal W and AntiClustAl. Our results demonstrate a very high conservation of the enzyme aminoacid sequence during evolution: this proves a very tight structure - function relationship, as it is to be expected for very ancient molecules performing critical biological functions. Less pronounced is the nucleotide sequence conservation, as it was foreseeable due to the species - specific codon use. It is noteworthy that evolutionary trees, drawn by using all the available data on SOD2 aminoacid and nucleotide sequences and both Clustal W and AntiClustAl, are comparable to those obtained through classical phylogenetic analysis.

## 7 Conclusions

In this paper we present AntiClustAl a new method to fast align a set of biosequences. The technique is based on a hierarchical clustering which yields a guide tree of then sequences and the align such sequences by a post order traversal of the tree. A successful biological application concerning the enzyme *Xenopus laevis* SOD2 shows the applicability of the method. The method compared with Clustal W shows a better running time together with a comparable alignment quality. The software is available on line at the following address: <http://alpha.dmi.unict.it/~ctnyu/>. We believe that randomized tournament techniques can be successfully used in other bioinformatics. This will be the subject of future research.

## References

- [1] M. Brudno, C. Do, G. Cooper, M. F. Kim, E. Davydov, NISC Sequencing Consortium, E. D. Green, A. Sidow, and S. Batzoglou. Lagan and multi-lagan: Efficient tools for large-scale multi-

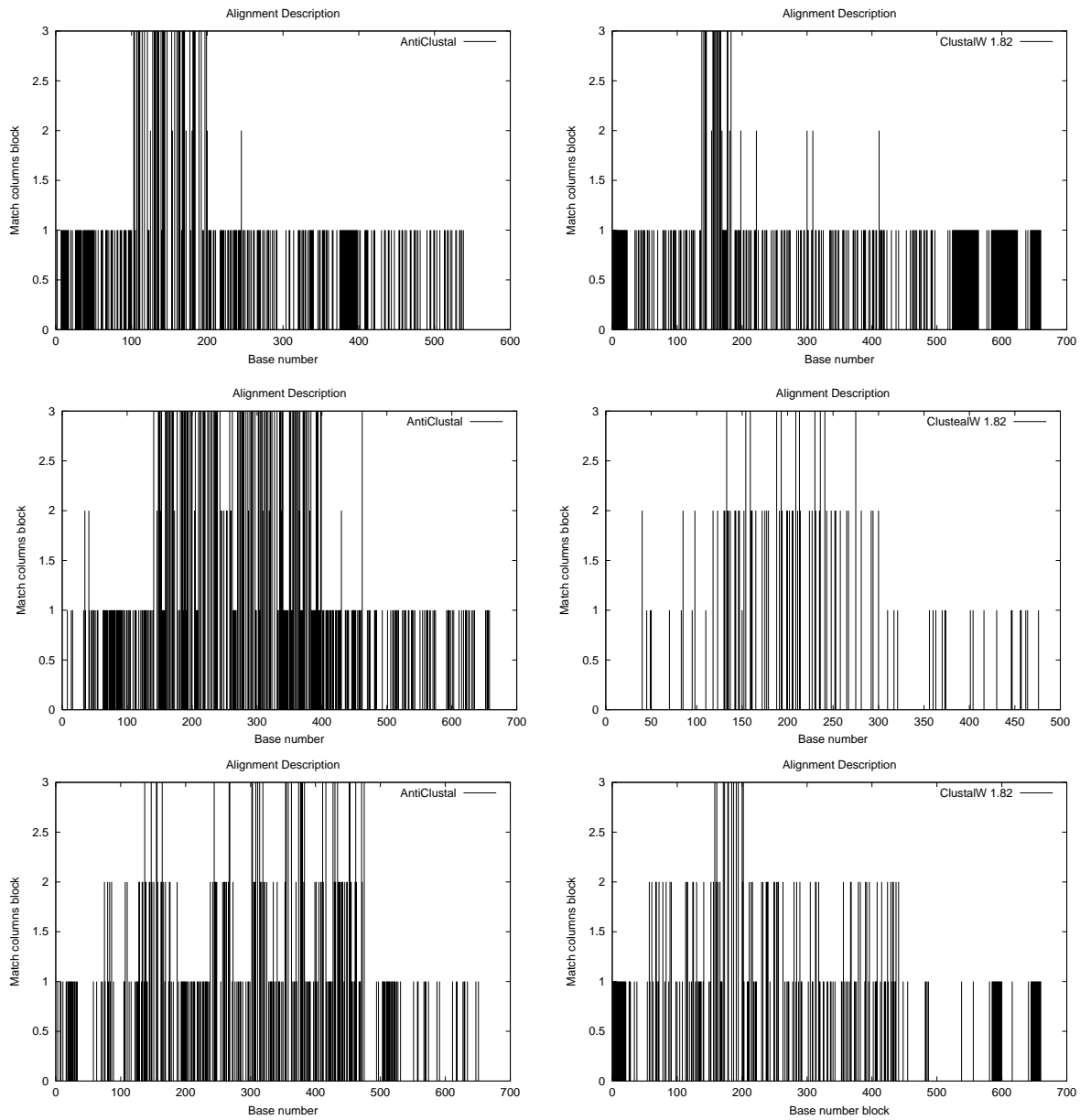


Figure 9: Alignment Comparison between ClustalW1.82 and AntiClustal with alpha-globins

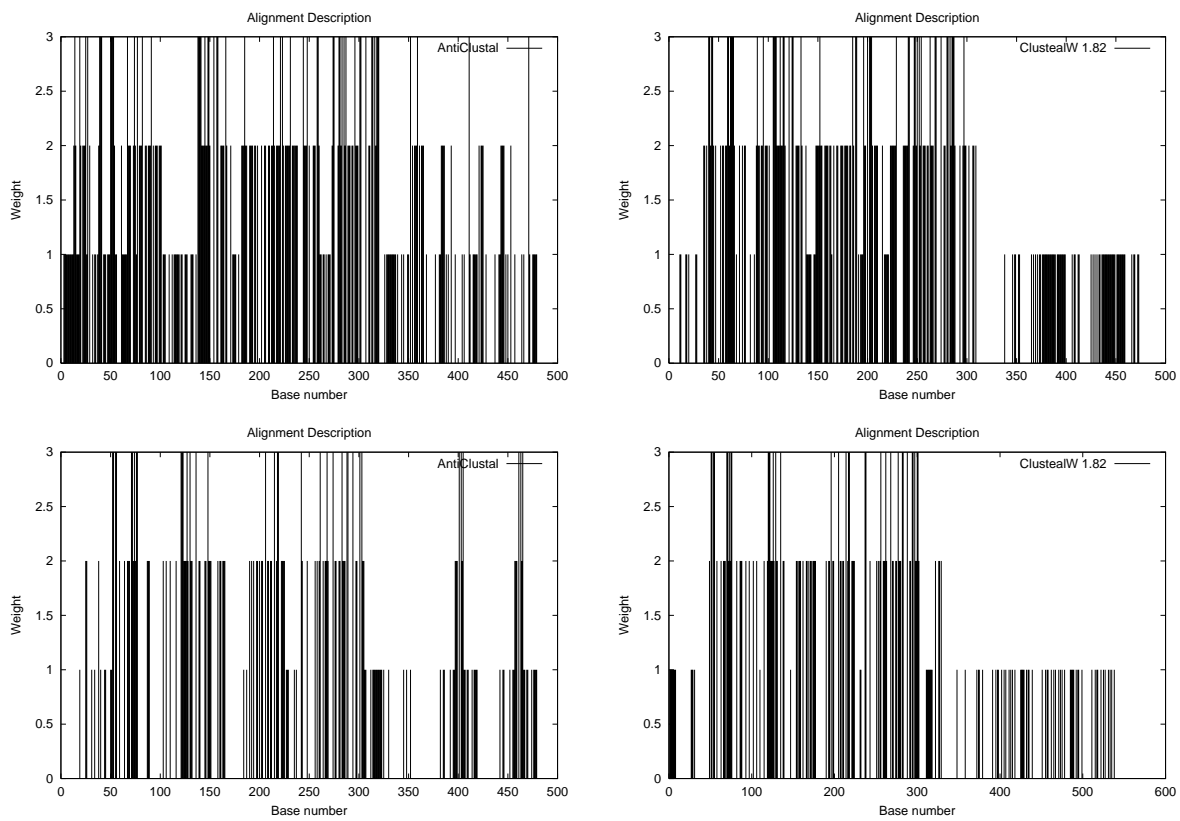


Figure 10: Alignment Comparison between ClustalW1.82 and AntiClustal with immuno-globulins

- ple alignment of genomic dna. *Genome Research*, in press.
- [2] D. Cantone, G. Cincotti, A. Ferro, and A. Pulvirenti. An efficient algorithm for the 1-median problem. *Technical Report University of Catania, Submitted*, 2003.
- [3] D. Cantone, A. Ferro, A. Pulvirenti, D. Reforgiato, and D. Shasha. Antipole indexing to support range search and  $k$ -nearest neighbor metric spaces. *Technical Report University of Catania, Submitted*, 2003.
- [4] H. Carrillo and D. Lipmann. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48:1073–1082, 1988.
- [5] E. Chavez, G. Navarro, R. Baeza-Yates, and J.L. Marroquin. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001.
- [6] M. Crochemore, G.M. Landau, and M. Ziv-Ukelson. A sub-quadratic sequence alignment algorithm for unrestricted scoring matrices. *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 679–688, 2002.
- [7] D. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 60:351–360, 1987.
- [8] W.M. Fitch and E. Margoliash. Construction of phylogenetic trees. *Science*, 155:279–284, 1967.
- [9] M. Gribskov, R. Luethy, and D. Eisenberg. Profile analysis. *Methods in Enzymology*, 183:146–159, 1989.
- [10] D. Gusfield. *Algorithms on Strings, Trees and Sequences*. Cambridge University Press, New York, 1997.
- [11] D.G. Higgins, J.T. Thompson, and T.J. Gibson. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acid Research*, 22:4673–4680, 1994.
- [12] D.G. Higgins, J.T. Thompson, and T.J. Gibson. Using clustal for multiple sequence alignments. *Methods in Enzymology*, 266:383–402, 1996.
- [13] B. Morgenstern, K. Frech, A. Dress, and T. Werner. Dialign: Finding local similarities by multiple sequence alignment. *Bioinformatics*, 14:290–294, 1998.
- [14] E.W. Myers and W. Miller. Optimal alignments in linear space. *CABIOS*, 4(1):11–17, 1988.
- [15] M. Purrello, C. Di Pietro, M. Ragusa, A. Pulvirenti, G. Pigola, V. Zimmitti, V. Di Pietro, T. Maugeri, G. Emmanuele, E. Modica, S. Travali, M. Scalia, , D. Shasha, and A. Ferro. In vitro and in silico cloning of xenopus laevis sod2 demonstrates very high aminoacid sequence conservation during evolution. *Submitted*, 2003.
- [16] H. Saitou and M. Nei. The neighbor-joining method: A new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, 4:406–425, 1987.