Incremental Multiple Sequence Alignment*

Marcelino Campos, Damián López, and Piedachu Peris

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
46071 Valencia (Spain)
{mcampos,dlopez,pperis}@dsic.upv.es

Abstract. This work proposes a new approach to the alignment of multiple sequences. We take profit from some results on Grammatical Inference that allow us to build iteratively an abstract machine that considers in each inference step an increasing amount of sequences. The obtained machine compile the common features of the sequences, and can be used to align these sequences. This method improves the time complexity of current approaches. The experimentation carried out compare the performance of our method and previous alignment methods.

Keywords: Grammatical inference, processing of biosequences, multiple alignment of sequences.

1 Introduction

Multiple alignment of biological sequences [1] is one of the commonest task in bioinformatics. Some applications of this task are: to find diagnostic patterns in order to characterize protein families; to detect or demonstrate homology between new sequences and existing families of sequences; to help predict the secondary and tertiary structures of new sequences; to suggest oligonucleotide primers for PCR; or as a essential prelude to molecular evolutionary analysis.

In order to perform an exact alignment, it is necessary to consider an n-dimensional space, where n denotes the number of sequences. There are some strategies used to avoid the high computational cost of the multiple alignment of sequences. One of the most successful strategies used is the so named *progressive alignment* (i.e. [2]). This approach considers evolutionary relations to build a phylogenetic tree. Following its branching order it is possible to align first the those most related sequences, gradually adding in more distant sequences/alignments. A pairwise alignment procedure is used in each alignment step. This approach can deal with alignments of virtually any number of sequences, obtaining good results.

Two main problems arise when this approach is used. The first one is due to the greedy nature of the method that does not guarantee that the global optimal solution would be obtained. The second problem consist on the choice of the alignment

* This work is partially supported by the Spanish Ministerio de Educacion y Ciencia, under contract TIN2007-60769, and Generalitat Valenciana, contract GV06/068.

L. Rueda, D. Mery, and J. Kittler (Eds.): CIARP 2007, LNCS 4756, pp. 604–613, 2007.

parameters. Usually, a distance matrix between symbols and two gap penalties are chosen: the opening and the extension of a gap. On the one hand there is no way to obtain a consensus matrix for every kind of sequences, on the other hand the gap penalty values are key to obtain good results when divergent sequences are to be aligned. Furthermore, in protein sequence alignments, gaps do not occur randomly (they occur more often between secondary structures than within).

Other successful approach aims to reduce the greedy effect of the progressive alignment method by considering a library of local and global alignments instead of a distance matrix [3]. Due to the fact that a given pair of symbols can be treated in many ways in the library, the score given to a pair of symbols is position dependent. This lead to a more flexible approach and a better behaviour, avoiding errors in early stages of the process.

Inductive Inference is one of the possibilities to tackle the problem of Automatic Learning. This approach uses a set of facts (training data) in order to obtain the most suitable model, that is, the model that compile better the features of the data. Once the inference process is finished, the model obtained is able to correctly process data that shares some common features with the training set. When the inductive process obtains a formal language as the model, then the approximation is known as *Grammatical Inference* (GI) [4,5,6,7,8,9].

The *Error Correcting Grammar Inference* algorithm (ECGI) [10] builds iteratively a finite automaton (learns a regular language) from the training data. Each step considers the automaton of the previous step (the first step it considers the empty automaton) and a new sample from the training data. The algorithm uses an edit distance algorithm to detect the set of operations of minimum cost needed to force the automaton to accept the sample. These operations are used to modify the automaton, adding new states and transitions in such a way that neither loops nor cycles are added (the resulting automaton accepts a finite language). This GI method has been successfully applied in some pattern recognition tasks [11,12].

In our work we used this algorithm to learn a language from biological sequences. Once the finite automaton is obtained, it is possible to extract an alignment using the automaton's accepting path of each sequence. This step is bounded by a polynomial because the automaton lacks of loops. Our approach improves the time complexity of current procedures.

This paper is organized as follows: section 2 introduces some notation and definitions as well as the ECGI algorithm. Section 3 explains the proposed alignment method and analyzes its time complexity. Section 4 shows the experimentation carried out and comparative results with current alignment methods. The conclusions and some lines of future work end the paper.

2 Definitions and Methods

2.1 Theoretical Concepts

Let a finite deterministic automaton be a system $A = (Q, \Sigma, \delta, q_0, F)$ where: Q is a finite set of states; Σ is a finite alphabet; $q_0 \in Q$ is the initial state; $F \subseteq Q$ is

the set of accepting or final states and $\delta : Q \times \Sigma \rightarrow Q$ is the set of transitions of the automaton. It is possible to change the definition of the transition function to allow multiple transitions from a state on the same symbol $\delta : Q \times \Sigma \rightarrow 2^Q$. Such an automaton is referred to as non-deterministic finite automaton. Both types of automata recognize the same class of formal language. Finite automata can be extended to process strings of symbols, to do so, the transition function has to be extended to consider strings. Let p, q be states in Q, let a be a symbol in Σ and w a string, in the following we show the extension for the non-deterministic case:

$$\begin{aligned} &\delta : Q \times \Sigma^* \rightarrow 2^Q \\ &\delta(q, \lambda) = q \\ &\delta(q, wa) = \bigcup_{p \in \delta(q,w)} \delta(p, a) \end{aligned}$$

A string w over Σ is accepted by a finite automaton A if and only if $\delta(q_0, w) \cap F \neq \emptyset$. The set of strings accepted by the automaton is denoted by $L(A)$. Let Σ^* be the set of words of any length over Σ. Let $L \subseteq \Sigma^*$ be a language over the alphabet. L is a regular language if and only if there exists a finite automaton A such that $L(A) = L$. Please, refer to [13] for further definitions.

2.2 The Error Correcting Grammar Inference

The Error Correcting Grammar Inference (ECGI) algorithm proposed by Rulot and Vidal [10] was originally designed to recognize isolated words. Nevertheless, due to its features, it has been used in many others pattern recognition tasks. The ECGI solve two basic drawbacks of grammatical inference when applied to pattern recognition tasks. First, these algorithms are usually extremely recursive, that is, they ignore the relative position of the different substructures of the training sample. Second, grammatical inference algorithms, usually, do not maintain position-dependent features of the strings, which are key in some tasks. The ECGI algorithm obtains a finite language that preserves the main common features of the samples together with this relative position.

ECGI algorithm is an iterative process. ECGI considers one new sample and the current automaton and finds the most similar string to the sample (applying a criterion of similarity between strings that take into account a distance measure). The algorithm detect those transitions with minimum edition cost to be added in order to accept the new sample. The search of the most similar string to the sample in the automaton is made by a standard Error-Correcting Syntax Analysis method (Viterbi). An example of run is shown in Figure 1.

The language inferred by the automaton with ECGI method contains the training samples, is finite (therefore regular) and is not deterministic. The automaton obtained is ambiguous and without cycles. Due to its heuristic nature, the inference algorithm obtains different result depending on the samples order. The algorithm has been successfully applied to syntactic pattern recognition

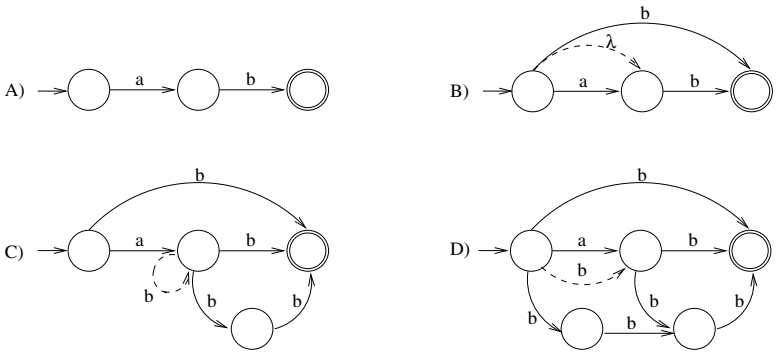

Fig. 1. Example of run of the original ECGI algorithm when the set $S = \{ab, b, abb, bbb\}$ is considered as the training sample. Dashed lines show error transitions. Note that the processing of the first string of the training set S returns the canonical automaton (see A). In order to avoid lambda transitions, the ECGI algorithm adds a transition to the next state (see B). Loops are avoided by the creation of a new state (see C), substitution are treated in the same way (see D). Note that, in the resulting automaton, all the incoming transitions to each state are labeled with the same symbol.

tasks [11,12], where the best performance of the algorithm is obtained when the longest samples are supplied first.

3 Incremental Alignment of Biosequences

3.1 Description of the Alignment Procedure

The method we propose for the multiple sequence alignment consist on two steps: the first one considers the set of sequences and obtains an automaton with a modified ECGI algorithm; the second step uses the learned automaton and the same set of sequences to construct the multiple alignment.

The Error-Correcting Syntax Analysis method used by the original ECGI algorithm considers three weights: to substitute, to insert and to erase a symbol. The modification we introduce allows the use a parameterizable distance matrix among the symbols and three gap-related penalties: to open, to extend and to close a gap. This modification of the analysis step does not add time complexity to the algorithm, and basically aims to change, in a somewhat biological way, the set of non-error transitions of the automaton obtained in each analysis. The use of such gap penalties is justified biologically and widely used by existing approaches. The alignments obtained this way have lesser and more concentrate gaps. Figure 2 shows the different behavior of the modified algorithm when the same sample is considered.

The alignment method we propose uses the set of sequences to align as the training set. Once inferred the automaton, the sequences are processed to obtain the accepting path in the automaton. We considered those states used by more

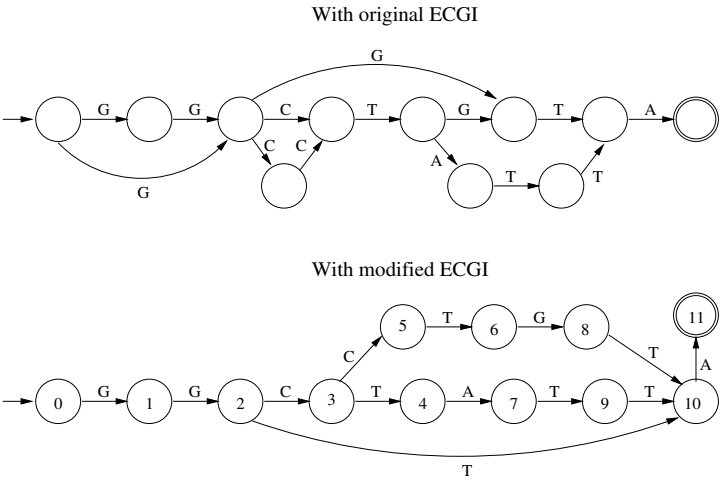

Fig. 2. Final automaton obtained by the original and modified ECGI algorithms. The same set of samples $S = \{GGCCTGTA, GGCTATTA, GGTA\}$ has been used. The optimal parameters were used to carry out this examples (see Section 4 for details). Note that the addition of gap penalties modifies the set of non-error transitions, and therefore, the final result.

```
PROGRAM: MULTIPLE SEQUENCE ALIGNMENT
INPUT:  STRINGS[]
OUTPUT:  ALIGNMENT

BEGIN

    AUT ←——— result of the ECGI algorithm using as input the strings into STRING[];
    PATHS[] ←——— set of paths obtained by Viterbi's analysis using the automaton AUT and STRINGS[];
    ALIGNMENT ←— aligment obtained using the set of paths PATHS[], the automaton AUT and STRINGS[];

END
```

Fig. 3. Multiple sequence alignment algorithm based on Grammatical Inference

than one sequence as synchronization points. This is justified because, each step, the error-correcting inference method looks for the best accepting path for the samples, adding the minimum cost transitions needed to accept the sample. The lack of loops allow to efficiently process the sequences in a parallel way, adding gaps when one synchronizing state is reached by some sequences but not all that use the state. The description of the algorithm is showed in Figure 3 and an example of multiple alignment in Figure 4.

In order to compare the performance of our approach, two alignment programs have been selected. The first one, Clustal-W [2], is probably the most widely used by the biological community. T-Coffee [3] is the second program considered. Briefly, T-Coffee could be seen as a refinement of the progressive approach to multiple alignment that avoids the use of a distance matrix. These two programs

	original ECGI	modified ECGI
string 1	GGCCTG.TA	GGCCTGTA
string 2	GG.CTATTA	GGCTATTA
string 3	.G...G.TA	GG....TA

Fig. 4. Alignment example using the automata from Figure 2. Considering the version that uses the modified ECGI, please note that the state 10 is a synchronizations point for all the strings. Note that the third string reaches this state with its third symbol. The other two strings need to analyze several symbols to reach the same state. Therefore, somewhat the third string "has to wait", and a gap is opened. The gap is closed when all the strings reach the synchronization state. Also note that the original ECGI automaton lead to an alignment that contains more gaps than the alignment obtained when the modified ECGI algorithm is used.

allow us to compare the GI approach in terms of computational complexity as well as their experimental behaviour.

3.2 Computational Complexity

In the following let n denote the number of sequences to align and let M denote the length of the longest sequence. The time complexity of Clustal-W algorithm is $\mathcal{O}(n^2M^2)$. The time complexity of T-Coffee is higher than the complexity of Clustal-W algorithm ($\mathcal{O}(n^2M^2) + \mathcal{O}(n^3M) + \mathcal{O}(n^3) + \mathcal{O}(nM^2)$).

The approach we propose needs $\mathcal{O}(n)$ steps to build the automaton and each one can be carried out in $\mathcal{O}(M^2)$, therefore, the automaton can be obtained with complexity $\mathcal{O}(nM^2)$. The second step implies the alignment of the sequences, and can be carried out with complexity $\mathcal{O}(nM)$. Therefore, the final time complexity of our alignment method is $\mathcal{O}(nM^2)$, therefore improving previous results.

4 Experimental Results

In order to carry out the experimentation, a benchmark database of RNA alignments was considered [14]. Structural alignment of RNA remains as an open problem despite the effort on the development of new alignment procedures to protein sequences. This dataset is divided into five subsets that take into account structural features. In their work, Gardner et al., compare the performance of several methods. Their results were the reason to select Clustal-W and T-Coffee as the two better methods.

From the five subsets of the database, one of them was thought to be untrustworthy, and was not considered. Further testing has shown the alignments of the set to be perfectly reliable and were considered in our work.

To assess the performance of the results, two different scores have been used [15]. The *sum-of-pairs score* (SPS) increases with the number of sequences correctly aligned. Therefore, it is useful to measure whether the program succeeds in aligning some of the sequences in an alignment. The *column score* (CS) is a

more strict parameter and it is a way to test whether the program aligns or not all the sequences properly.

Given an alignment of N sequences and M columns, let $a_{i1}, a_{i2}, \ldots, a_{in}$ denote the symbols on the ith column. p_{ijk} is defined such that $p_{ijk} = 1$ if symbols a_{ij} and a_{ik} are aligned with each other in the reference alignment and $p_{ijk} = 0$ otherwise. Let the score S_i be defined as follows:

$$S_i = \sum_{j=1}^{N} \sum_{\substack{k=1 \\ k \neq j}}^{N} p_{ijk}$$

let M_r denote the length of the reference alignment and S_{ri} the score of the ith column for the reference alignment. The SPS for the alignment is then:

$$SPS = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}}$$

For any given alignment described as above, let $C_i = 1$ if all the symbols in the ith column are aligned in the reference, otherwise let $C_i = 0$. Then, CS for the alignment is obtained as follows:

$$CS = \frac{\sum_{i=1}^{M} C_i}{M}$$

Clustal-W and T-Coffee software were downloaded from the European Bioinformatics Institute (*http://www.ebi.ac.uk/*). The experiments were run with the DNA default parameters for each program because they are reported as the most suitable ones. We tested both the basic ECGI algorithm and the modified one. The same alignment scheme explained in Section 3 was followed in any case.

All the parameters used in the alignment step of our approach were empirically set to: an identity distance matrix with substitution value set to 6; a gap open penalty set to 10; a gap extension penalty set to 6.66; and a closing gap penalty set to 0.5. The original version of the ECGI algorithm showed poor performance (results not showed). The main reason of this behaviour was the high amount of gaps introduced in the alignment, mainly due to the lack of gap penalties.

Experiments were carried out for each alignment group in the dataset. From the data shown we can consider that, no matter which group is considered, there is no substantial difference in the behaviour of the methods tested. Three homology levels were studied: low homology for sequences below the 50% of identity, medium homology for sequences between 50% and 70% of identity, and high homology for sequences over 70% of identity. No relevant difference in the methods' performance was observed. Finally, Figure 5 shows the global results. From the data obtained we conclude that all the three methods behave in the same way. Table 1 summarizes all the results.

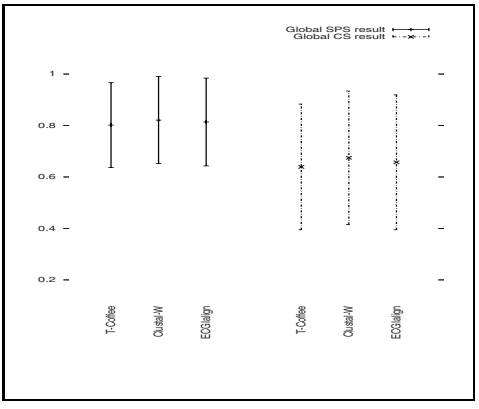

Fig. 5. Global experimental results

Table 1. Summary of the experimental results

		T-Coffee		Clustal-W		ECGIalign	
		SPS	CS	SPS	CS	SPS	CS
Results by group	G1	0.698	0.506	0.700	0.508	0.682	0.476
	G2	0.895	0.773	0.927	0.836	0.906	0.800
	G3	0.835	0.679	0.856	0.719	0.836	0.679
	G4	0.802	0.665	0.849	0.731	0.868	0.755
	G5	0.782	0.582	0.779	0.593	0.780	0.583
Results by homology	Low	0.735	0.536	0.749	0.562	0.746	0.549
	Med	0.868	0.736	0.903	0.801	0.892	0.780
	High	0.910	0.831	0.928	0.863	0.929	0.862
Global results		0.801	0.638	0.820	0.674	0.813	0.656

5 Conclusions and Future Work

Multiple alignment of biological sequences is one of the commonest task in bioinformatics and have several important applications. Structural alignment of RNA remains as an open problem despite the effort on the development of new alignment procedures to protein sequences. In this work, we address this task using only the sequences of the molecules to align.

Inductive inference is one of the possibilities to tackle the problem of Automatic Learning. In our work we used a grammatical inference algorithm and biological information to learn a finite automaton. This automaton is used to obtain an alignment using the each sequence accepting path in the automaton.

It is important to note that the time complexity of our approach to multiple sequence alignment improves previous results. Our method achieves the same performance but reduces by one degree the time complexity of previous approaches ($\mathcal{O}(nM^2)$ instead $\mathcal{O}(n^2M^2)$ where n denotes the number of sequences to align and M denotes the length of the longest sequence).

Several lines of work remain open and could lead to improve our results. As noted above, ECGI algorithm obtains different automata when the training set is ordered in different ways. Therefore it should be possible to improve the results by ordering the samples as a preprocessing of the training set. Nevertheless this preprocessing would lead to an increasing of the time complexity, and it has to be studied whether the performance worth the increased complexity.

Another important feature of the grammatical inference algorithms is that the more data available, the best results obtained. Therefore, it has to be studied whether the more sequences considered, the better alignment results obtained.

One of the drawbacks of previous alignment methods is the greedy behaviour of the approach that makes impossible of change first stages alignments. It can be argued that our approach can also produce bad first-stage decisions. In order to smooth this greedy behaviour, it should be possible to take advantage of the ambiguity of the automata by using stochastic automata. In this way, those better accepting paths in the automata would be selected, leading to better alignment results.

References

1. Notredame, C.: Recent progresses in multiple sequence alignment: a survey. Pharmacogenomics 3(1), 1–14 (2002)
2. Thompson, J., Higgins, D., Gibson, T.: 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(22), 4673–4680 (1994)
3. Notredame, C., Higgins, D., Heringa, J.: T-coffee: a novel method for fast and accurate multiple sequence alignment. J. Mol. Biol. 302, 205–217 (2000)
4. Fu, K., Booth, T.: Grammatical inference: Introduction and survey - Part I. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(3), 343–359 (1975)
5. Fu, K., Booth, T.: Grammatical inference: Introduction and survey - Part II. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(3), 360–375 (1975)
6. Angluin, D., Smith, C.: Inductive inference:Theory and Methods. Computing Surveys 15(3), 237–269 (1983)
7. Miclet, L.: Grammatical inference. In: Syntactic and Structural Pattern Recognition. Theory and Applications. Series in Computer Science, vol. 7, pp. 237–290 (1990)
8. Sakakibara, Y.: Recent advances of grammatical inference. Theoretical Computer Science 185, 15–45 (1997)
9. Sakakibara, Y.: Grammatical inference in bioinformatics. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(7), 1051–1062 (2005)
10. Rulot, H., Vidal, E.: An efficient algorithm for the inference of circuit-free automata. In: Syntactic and Structural Pattern Recognition. NATO Asi Series, pp. 173–184 (1988)
11. Prieto, N., Vidal, E.: Learning language models through the ecgi method. Speech Communication 11, 299–309 (1992)
12. Vidal, E., Rulot, H., Valiente, J.M., Andreu, G.: Application of the error-correcting grammatical inference algorithm (ECGI) to planar shape. In: Grammatical inference: theory, applications and alternatives. vol. IEE. Digest No: 1993/092 (1993)

13. Hopcroft, J., Ullman, J.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley Publishing Company, Reading (1979)
14. Gardner, P., Giegerich, R.: A comprehensive comparison of comparative rna structure prediction aproaches. BMC Bioinformatics 5 (2004)
15. Thompson, J., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. Nucleic Acid Research 27(13), 2682–2690 (1999)