A Double Combinatorial Approach to Discovering Patterns in Biological Sequences

M.-F. Sagot [1,2] A. Viari [1]

[1] Atelier de BioInformatique
CPASO - URA CNRS 448
Section de Physique et Chimie de l'Institut Curie
11, rue P. et M. Curie 75005 - Paris - FRANCE

[2] Institut Gaspard Monge
Université de Marne la Vallée
2, rue de la Butte Verte 93160 - Noisy le Grand

email: sagot@radium.jussieu.fr

Abstract

We present in this paper an algorithm for finding degenerated common features by multiple comparison of a set of biological sequences (nucleic acids or proteins). The features that are of interest to us are words in the sequences. The algorithm uses the concept of a model we introduced earlier for locating these features. A model can be seen as a generalization of a consensus pattern as defined by Waterman [42]. It is an object against which the words in the sequences are compared and which serves as an identifier for the groups of similar ones. The algorithm given here innovates in relation to our previous work in that the models are defined over what we call a weighted combinatorial cover. This is a collection of sets among all possible subsets of the alphabet Σ of nucleotides or amino acids, including the wild card $\{\Sigma\}$, with a weight attached to each of these sets indicating the number of times it may appear in a model. In this way, we explore both the space of models and that of alphabets. The words that are related to a model defined over such a combinatorial cover, and thus considered to be similar, are then the ones that either belong to the model or present at most a certain number of errors with a nearest element of it. We use two algorithmic ideas that allow us to deal with such double combinatorics, one concerns a left-to-right minimality of the sets composing a model, the other involves making a sketch of the solution space before exploring it in detail.
keywords : multiple comparison, weighted combinatorial cover, wild card, model, degenerated feature, left-to-right minimality of sets, sketch of solution space, DNA, protein.

1 Introduction

One important goal in computational molecular biology is finding features that are common to a set of nucleic acid or protein sequences. A vast literature can be found that either addresses the subject directly [4] [5] [12] [13] [15] [16] [17] [18] [19] [20] [21] [26] [28] [29] [30] [34] [35] [37] [38] [40] [41] [42] or proposes algorithms for the multiple alignment of sequences that start by looking for such common features [3] [14] [27] [31] [39] or from which these features may be subsequently extracted [3] [7] [8] [10] [32] [33] [36]. In the case of sequences, it has been traditional to consider that the features of interest correspond in general to words, that is, to contiguous series of symbols in the sequences. In this work, features and words are then considered to be synonymous.

The motivations behind this search for common features are diverse although they all reflect the same underlying assumption: that what is (more or less) well-conserved among a set of objects upon which act factors of variability may be biologically important. Given a set of such sequences, the features that interest us are then in general those that appear a great number of times, although the infrequent features may also be important [5] [11].

The difficulties behind this problem of finding common features or words are of various nature. The first one comes from the fact that perfect conservation, especially at the sequence level, is not necessary for preservation of biological role, and so is seldom observed. What is more often conserved instead are physico-chemical properties. The common features we shall be looking for are thus not identical words but approximately similar ones. This first difficulty brings up a second one which is the necessity for defining what is meant by a set of words being similar, how is this similarity going to be characterized, which properties should it verify. This definition may vary depending on the kind of feature one is interested in, functional, structural or evolutionary. A final difficulty comes from the combinatorial nature of the problem itself since the best way for finding even weakly similar features is by doing multiple comparisons. The algorithms that have been proposed so far have in general found a way around these difficulties either by more or less severely restricting the kinds of features one may look for (for instance, by fixing their size a priori or by equating similarity with identity), or they have resorted to heuristics to find them.

Yet some of the papers mentioned above have tried to address the issues we just raised in a flexible but precise way (notably [18] [29] [38] [39] [40] [41] [42] which are the ones most related to our own approach) and this has been our main aim too. This work places itself inside a series of previous ones [22] [24] [25] we presented recently on this same subject of giving various rigorous definitions of the notion of similarity between words in sequences and of providing exact ways for finding similar features based on any of these definitions. Different mathematical concepts were used for this, among them that of a model. Although what we are comparing are concrete objects, sequences, and so the features we are looking for correspond as we saw to words in these sequences, we have shown in [23] that underlying all such searches is this idea of an object - a model - against which the words in the sequences are in fact being implicitly or explicitly compared. This object is external to the sequences, and although in its simplest form it is itself a word, it may be defined in other more complex ways. Whatever its definition, a model corresponds to a maximal set of words in the sequences that are related to it under some definition of similarity. From this definition one may then induce a relation of similarity between the words themselves. Models have been used previously, and to our knowledge Waterman was the first one to introduce them under the name of consensus patterns [38] [39] [40] [41] [42].

The general line of thought we followed in our previous papers is pursued in this work, in particular we henceforward talk always of exhaustively looking for models present in a set of sequences with the understanding that this effectively corresponds to exactly looking for common similar words in these sequences. We call these words that are all related to a same model M the occurrences of M. It is important to observe that, as in the previous papers, no a priori is made on which common features, resumed into models, are present in a set of such sequences. Given a definition for such models and for the similarity between them and words in the sequences, the space of all these models is explored in a combinatorial way. The various economies that can be made during this exploration so as to keep it as efficient as possible never eliminate a valid solution.

Yet some important new issues are presented in this paper in relation to our previous work.

The first one concerns introducing a different kind of flexibility in the definition of similarity between a model and its occurrences. Indeed, in all our previous papers, the occurrences of a model corresponded to words where all positions were significant for the signal, function or structural feature the model was trying to capture. As has been pointed out by Posfai [19], Bashford [2], Rooman [20] [21], Smith [29] and Neuwald [18] among others, this is not always the case with biologically interesting features as these often correspond to models that may contain non specific positions, that is, positions where conservation is unimportant. Of course, all positions in a model cannot be indifferent so a constraint is added that bears on the maximum number of non specific positions it may present. Working with this concept of model is different from working with a definition of similarity between models and words that allows for errors as we use in [22] and [25]. There is no idea of an error here (although errors could also be permitted as we show later): some of the positions in a model of interest are simply considered to be indifferent for the biological role played by that model.

One can go even further. Indeed, as observed by Waterman [5] and Karlin [11] among others, there is no way to know beforehand which choice of alphabet will be the good one for revealing common features. It seems therefore interesting to be able to work with different alphabets, possibly simultaneously. In this paper then, we present an algorithm where the alphabet of models is also explored in a combinatorial way. Models are thus defined as products of sets as was done for proteins in [25] but these sets may be taken now from all possible subsets of the alphabet Σ of nucleotides or amino acids. This includes as mentioned above the set composed of the whole alphabet itself (corresponding to the wild card symbol). Models are thus defined over $\mathcal{P}(\Sigma)$. As observed previously, concerning the set $\{\Sigma\}$, a constraint is imposed on the number of times it may appear in a model. A constraint of the same kind can also be imposed on the number of times any of the proper subsets of the alphabet of nucleotides or amino acids appears in the models that will interest us. This will be necessary in practice with proteins, but can also be used for exploring unusual characteristics of any sequences. The new alphabet obtained once these constraints have been imposed is called a weighted combinatorial cover WCC of Σ.

It is obvious that this approach greatly increases the complexity of comparing a set of sequences. Two further innovations this paper introduces concern therefore algorithmic ideas that permit us to be more efficient in practice, in some cases dramatically so, in the search for models. Although both algorithmic ideas exploit the nature of the data we are dealing with, they may be of a more general applicability. The first idea introduces a left-to-right minimality of the sets composing a model and the second idea shows a technique for exploring a search space that consists in first making a sketch of the solution space.

Finally, weighted combinatorial covers of a more restricted nature where the wild card is the only set whose appearance in a model is constrained can be combined with errors as in [22] and [25] to produce a version of the algorithm that has started showing promise for locating very degenerated features in both nucleic acid and protein sequences.

The final version of the algorithm with no errors allowed comprises two phases and its time complexity is bounded over by $O(n.N.k.2^k + m.k.p)$ where n is the average length of the sequences, N is their number, k is the length of the models searched for, m is a value related to the number of solutions found in the first phase that is in general much smaller than $n.N.2^k$ and p is the number of sets in the cover (so that $p \leq 2^{|\Sigma|}$).

The time complexity of the algorithm with a more restricted cover and with errors permitted is bounded over by $O(n.N.k^{2e+1}.g^k.(\mid \Sigma \mid + 1)^{e+1})$ where e is the maximum number of errors (substitutions, deletions and insertions) authorized and g is related to the degree of non transitivity of the cover.

Section 2 is devoted to a formal statement of the problem. For clarity, we give the more general case of a fully combinatorial cover as is used for treating nucleic acid sequences and with no errors allowed. The algorithm is gradually presented in sections 3 to 5, first the straightforward approach, then each of the ideas that result in an improvement of the algorithm's performance. In section 6, we introduce the algorithm with a restricted weighted combinatorial cover where errors are permitted. Finally, section 7 shows performance results and a brief illustration of the possible application of both algorithms to nucleic acids and proteins.

2 The Weighted Combinatorial Cover Problem

Let Σ be the alphabet of nucleotides, that is, $\Sigma = \{A, C, G, T \text{ or } U\}$. Let sequence s be an element of Σ^*. An element u of Σ^n for $n \geq 1$ is called a word in s if $s = xuy$ with $x, y \in \Sigma^*$.

The models we are interested in constructing in this paper are defined as:

Definition 2.1 *A model M of length k is an element of $(\mathcal{P}^+(\Sigma))^k$ where $\mathcal{P}^+(\Sigma) = \mathcal{P}(\Sigma)\backslash\{\emptyset\}$. We note $\mid M \mid$ the length of M.*

As we said, a model M is an object against which the words in the sequences are compared.

Definition 2.2 *Given a sequence $s \in \Sigma^*$ and a model $M \in (\mathcal{P}^+(\Sigma))^k$, M is said to be present (or to occur) in s at location i if the word u of length $\mid M \mid$ starting at position i in s is such that $u \in M$. u is called an image of M and we say that (i, u) is an occurrence of M in s.*

We note $Occ(M)$ the set of all occurrences of M in s, that is:

$$Occ(M) = \{(i, u = s_i ... s_{i+|M|-1}) \mid u \in M\}$$

Example 2.1 *Let $s = ACGTACCTT$. Then $M = \{A\}\{C\}\{C,G\}\{T\}$ occurs at positions 0 and 4 in s.*

Of course, if $M = \{\Sigma\}^k$ for $k \geq 1$ then, trivially, M occurs at all positions $i \in \{0, ..., \mid s \mid -k\}$ of s. Such models are obviously uninteresting, so we define models as elements of what we call a weighted combinatorial cover instead where:

Definition 2.3 *A weighted combinatorial cover WCC of Σ is defined by:*

$$WCC = \{(S, w_S) \mid S \in \mathcal{P}^+(\Sigma) \text{ and } w_S \text{ is a non-negative integer}\}.$$

Definition 2.4 *Given a weighted combinatorial cover WCC of Σ, a model $M = S_1 ... S_k$ is an element of WCC^k if, for all $1 \leq i \leq k$, $S_i \in \mathcal{P}^+(\Sigma)$ and the number of times S_i appears in M is no greater than w_{S_i}.*

Example 2.2 *Let $\Sigma = \{A, C, G, T\}$ and:*

$$WCC = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 1)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}$$

Then we have:
$\{A,C\}\{A,G\}\{A,T\}\{C,G\}\{C,T\}\{G,T\} \in WCC^6$
but for instance:
$\{A,C\}\{A,C\}\{A,T\}\{C,G\}\{C,T\}\{G,T\}$
and
$\{A,C\}\{A,G\}\{A,T\}\{C,T\}\{A,C,G,T\}\{A,C,G,T\}$
are not elements of WCC^6.

From now on, given a weighted combinatorial cover WCC, we call an element M of $(\mathcal{P}^+(\Sigma))^k$ a model only if $M \in WCC^k$.

As we mentioned in the introduction, the models that will interest us are those that appear a certain number of times in the set of sequences we are comparing. We say of such models that have occurrences in at least q different sequences of the set that they satisfy a quorum constraint. The problem we want to solve can therefore be stated in this way:

Weighted Combinatorial Cover Problem *Given an alphabet Σ, a weighted combinatorial cover WCC of Σ, a set $\{s1, s2, ..., sN\}$ of sequences over Σ, that is of elements of Σ^*, and a quorum constraint q that is a constant between 2 and N, we propose to solve the following problems:*

1. *find all models $M \in WCC^k$ that are present in at least q of the sequences of the set;*
2. *find the greatest length k_{max} for which there exists at least one model $M \in WCC^{k_{max}}$ that is present in at least q of the sequences of the set, and solve problem 1 for $k = k_{max}$.*

Example 2.3 *Let $\Sigma = \{A, C, G, T\}$ and:*

$$WCC = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 0) \end{array} \right\}$$

Given the three strings:

$s1 = GAGTCAGTAC\underline{TACACCGTAATCAT}ATACCGG$

$s2 = CCGATTACGACCATA\underline{TACAGCATTATAAT}GC$

$s3 = GACCCAGG\underline{TACAGCGGAATAAC}AACCTTTAC$

then the longest model found in all three strings is the model:

$\{T\}\{A\}\{C\}\{A\}\{C,G\}\{C\}\{A,G\}\{G,T\}\{A,T\}\{A\}\{T\}\{A,C\}\{A\}\{C,T\}$

at positions 10, 15 and 8 respectively (positions in strings are counted from 0).

In practice, looking for models present in at least q of the sequences of the set $\{s1, s2, ..., sN\}$ can be done by searching for the models present at least q times in the string $s = s1s2...sN \in \Sigma^*$, with a constraint imposed on the positions of the occurrences of each model so that there are at least q of them that correspond to different sequences of the set. Models that solve either problem 1 or 2 are called valid models.

We show next an algorithmic solution to these problems. The algorithm is introduced in a gradual way. We start by showing a rather straightforward approach, then present in sections 4 and 5 two ways to improve that first approach to produce a final version of the algorithm that is in practice more efficient than the first one.

3 The Algorithm: a First Approach

3.1 Main Idea

The main idea for the algorithm is based on the observation that models, and their sets of occurrences, can be constructed by recurrence. This comes from the following lemma:

Lemma 3.1 *Let $s \in \Sigma^*$, $X \in \Sigma$ and i be a position in s. Let $WCC = \{(S, w_S)\}$ be a weighted combinatorial cover of Σ and $M = M'S \in WCC^k$. Then:*

$$(i, u = u'X) \in Occ(M = M'S)$$
$$\Longleftrightarrow$$
$$(i, u') \in Occ(M'),\ X \in S,\ (p(S, M') + 1) \leq w_S$$

where $p(S, M')$ = number of times the set S appears in model M'.

This lemma gives a simple way to construct the set $Occ(M)$ from that of $Occ(M')$ where $\mid M' \mid = \mid M \mid - 1$. The set $Occ(M)$ for $\mid M \mid = 1$ and $w_M \geq 1$ is obtained by simply sliding along sequence s and placing in $Occ(M)$ all elements (i, s_i) for which $s_i \in M$. Obviously, if $w_M = 0$, that is, if the set M is not permitted in the models, then $Occ(M) = \emptyset$.

The construction of the models is then straightforward and can be seen in terms of the traversal of a tree branching out into at most p branches at each node where p is the number of sets S in the cover for which $w_S > 0$. Each node of the tree is labeled by the name of a model M and contains a bucket with the positions of the occurrences of M. A branch links two nodes M and M' if $M = M'S$ for a set $S \in WCC$ and is therefore labeled by S. Applying the lemma to a model means elongating it of one unit to the right and verifying which of the correspondingly elongated occurrences still belong to the extended model M. This represents also going down the tree of one level. Of course, we can only go down a branch labeled by a set S for which $(p(S, M') + 1) \leq w_S$. Furthermore, M will be a valid model only if $Occ(M)$ further verifies the quorum constraint q.

3.2 Complexity

If the quorum q is not taken into account (that is, if $q = 1$) and if $w_S = \infty$ for all $S \in \mathcal{P}^+(\Sigma)$, the time complexity of this first version of the algorithm is majored by $O(n.N.k.g^k)$ where n is the average length of the sequences $s1, ..., sN$, k is the length of the models sought and g is the maximum number of sets of the cover WCC an element of Σ may belong to.

This comes from the fact that, given a position i in s, the number of models M' of length $k - 1$ that the pair (i, u) with $\mid u \mid = \mid M' \mid$ can be an occurrence of is majored by g^{k-1}. Each such model

M' on its turn can have at most $n.N$ occurrences. This means that the total number of occurrences present in the buckets of all nodes at level $k-1$ of the tree is at most $n.N.g^{k-1}$, and so this number is at most $n.N.g^k$ at the next level. Although in theory each node at level $k-1$ could be split into p branches, the total number of branches linking the nodes at level $k-1$ with those at level k cannot exceed $n.N.g^k$. The idea is then to split each node not p times, corresponding to one for each set in the cover, but only the number of times required to account for the sets whose symbols are actually present in the sequences at the next positions of the occurrences. Using this idea, one can get to level k with at most $n.N.g^k$ operations. The complete recurrence takes therefore $n.N.k.g^k$ operations.

Now in the case of nucleic acids, $g = 8$, which is big.

In practice though, w_S is not ∞ at least for $S = \{\Sigma\}$. More importantly, taking quorum q into consideration may considerably reduce the actual complexity of the algorithm by reducing the size of the search space we have to explore. This is shown next.

3.3 Pruning the Search Space

We saw in section 3.1 that solving the general problem for a fixed length k for instance, that is developing the recurrence relation of lemma 3.1, corresponds to performing a walk along the tree of models and to listing the content of all buckets at depth k of the tree. Now taking quorum q into account, it is obvious that if a model M' does not verify it, no model M having M' as its proper prefix can verify it either. The walk along the branch that passes through node M' can therefore stop at that node. The complexity given previously corresponds to a traversal of the tree of all possible models present at least once in the sequences, the complexity of the application of the algorithm to an actual biological problem corresponds to the traversal of the tree of valid models, that is to that of a pruned version of the tree of all possible models which can be quite sparser than the latter.

So the average time complexity can be much smaller than the value given previously. It still remains too big and the next sections are devoted to showing two ways in which this algorithm can become more efficient in practice, though still exhaustive and exact.

4 Introducing Left-To-Right Minimality of Sets

4.1 Main Idea

Let us consider the following example:

Example 4.1 *Let* $q = 3$,
$s1 = ACAA$
$s2 = CAGA$
$s3 = AGAA$
$s4 = CTGA$

and:

$$WCC = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 1)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, 1) \end{array} \right\}.$$

Then $M_1 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A, T\}$ *and* $M_2 = \{A, C\}\{A, C, G, T\}\ \{A, G\}\{A\}$ *are both valid models of length 4. Yet model* M_1 *brings no new information in relation to model* M_2 *since* $Occ(M_1) = Occ(M_2)$ *and* $M_2 \subseteq M_1$ *(where* $(M_2 = \prod_{i=1}^{k} S_2^i) \subseteq (M_1 = \prod_{i=1}^{k} S_1^i)$ *if* $S_2^i \subseteq S_1^i$ *for* $1 \leq i \leq k$*).*

That is true also of all models M_1' *that have model* M_1 *as a proper prefix: the information they contain is superfluous in relation to the one brought by the corresponding models* M_2' *of same length that have* M_2 *as proper prefix. No information is therefore lost if the subtree of the tree of models having* M_1 *as root is not traversed.*

Observe that the previous example is very different from the following one:

Example 4.2 *Let WCC and q be as before and now:*

$s1 = ACAA$
$s2 = CAGA$
$s3 = AGAA$
$s4 = ATGA$

Here $M_1 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A\}$, $M_2 = \{A\}\{A, C, G, T\}\{A, G\}\{A\}$ and $M_3 = \{A\}\{A, C, G, T\}\{A, G\}\{A, T\}$ are all valid models of length 4 but where M_3 is superfluous in relation to M_2 since $Occ(M_3) = Occ(M_2)$ and $M_2 \subseteq M_3$, the same does not hold of model M_1 in relation to M_2. The reason is that, although as product of sets, $M_2 \subseteq M_1$, we have $Occ(M_1) \neq Occ(M_2)$.

Models M_1 and M_2 may therefore lead to different final solutions that are both valid and the tree of models may not be pruned at the node labeled by any of them.

What these examples show is that if, given two valid models M_1 and M_2, the following two conditions are simultaneously verified:

c1 $Occ(M_1) = Occ(M_2)$

c2 $M_2 \subseteq M_1$

then the information brought by model M_1 is superfluous in terms both of product of sets of the cover and of occurrences. We can throw it away, and prune the tree of models at the node labeled by its name. In other words, where at a given step in the algorithm, the sets of occurrences of two models are found to be identical, we need to keep only the one that corresponds to a product of minimal sets in relation to the other. This can represent a further economy of time (and space).

In practice though, the two conditions given above are not the ones we actually verify. Instead, what is checked is whether, given two models $M_1 = MS_1$ and $M_2 = MS_2$, the following is true:

c3 $Occ(M) \cap Occ(S_1) = Occ(M) \cap Occ(S_2)$ (this is the same as c1 written in a different way)

c4 $S_2 \subseteq S_1$

This means that the minimality of sets in a model is always related to the last sets $S \in \mathcal{P}^+(\Sigma)$ that are successively concatenated to a same model M, it is never checked against all models. Clearly then, c3 and c4 are weaker conditions than c1 and c2. The following is an example of this:

Example 4.3 *Let WCC be as in example 4.1, $q = 3$ and:*

$s1 = ACAA$
$s2 = CAGT$
$s3 = AGAA$
$s4 = ATGA$

Then $M_1 = \{A, C\}\{A, C, G, T\}\{A, G\}$ and $M_2 = \{A\}\{A, C, G, T\}\{A, G\}$ are both valid, non superfluous models of length 3 and so both are kept. Models $M_3 = \{A\}\{A, C, G, T\}\{A, G\}\{A\}$ and $M_4 = \{A, C\}\{A, C, G, T\}\{A, G\}\{A\}$ of length 4 are also valid but M_4 is superfluous since $Occ(M_3) = Occ(M_4)$ and $M_3 \subseteq M_4$. In our implementation though both are kept because $M_3 = M_2\{A\}$ and $M_4 = M_1\{A\}$ and $M_2 \neq M_1$. So the minimality of M_3 in relation to M_4 cannot be verified.

Checking conditions c3 and c4 is necessary because models are constructed in a depth-first way and so verification of conditions c1 and c2 is impossible without adding to the complexity of the algorithm as it involves comparisons between all models of a given length.

The property that is verified by conditions c3 and c4 is then what we call a left-to-right minimality. We can state also that if a given model M is kept, all smaller models in lexicographic order of cardinality of sets that belong to WCC and verify the quorum are kept too. This is the case of M_4 and M_3 in example 4.3.

The question now is, how can we verify conditions c3 and c4 above in an efficient way?

Let M' be the model we are in the process of extending to the right by one unit. Let $Occ(M')$ be the set of its occurrences and $S = \{s_{i+|M'|} \in \Sigma \mid i \in Occ(M')\}$. In other words, S is the set of symbols of Σ that comes after each of the occurrences of M' in s.

Then the models $M = M'S'$ for $S' \in WCC$ that must be kept are those for which:

let M' be the model we are trying to extend of one unit to the right;
let $S = \{s_{i+|M'|} \in \Sigma \mid i \in Occ(M')\}$;
let Sup(S') be all the sets S'' such that $S' \subset S''$ and $\mid S'' \mid = \mid S' \mid + 1$;
MustTrySet = $\emptyset$;

for all sets $S' \in WCC$ taken by <u>increasing</u> cardinality
    if ($S' \subseteq S$)
        if $((p(S', M') + 1) \leq w_{S'})$
            $M = M'S'$ is a valid model;
        else
            MustTrySet = MustTrySet $\cup$ Sup(S');
    else if ($S' \in$ MustTrySet)
        if $((p(S', M') + 1) \leq w_{S'})$
            $M = M'S'$ is a valid model;
        else
            MustTrySet = MustTrySet $\cup$ Sup(S');

Figure 1: Verification of conditions c3 and c4.

1. $(p(S', M') + 1) \leq w_{S'}$
   <u>and</u>
2. (a) $(S' \subseteq S)$
   <u>or</u>
   (b) S' is a <u>smallest</u> set such that $S \subset S'$ and there exists $S'' \subseteq S$ with $(p(S'', M') + 1) > w_{S''}$.

Of course, $M'S'$ must further verify the quorum constraint. Observe also that there may be more than one set S' verifying 2(b).

4.2 Algorithm

An idea of the algorithm corresponding to an implementation of the two points above is given in figure 1.

4.3 Complexity

If the data structures Sup(S) for all S in the cover and MustTrySet are implemented as bitwise vectors and the union and membership operations are bitwise, then the time complexity for verifying conditions c3 and c4 is $O(p)$. It is therefore linear in the number of sets in the cover. The overall complexity is then bounded over by $O(n.N.k.g^k.p)$. Although this is worse in theory than for the straightforward approach, in practice this is not so, the reason being that less models are produced and the search space is sparser. Experiments have shown (results not given here) that the time factor gained in the execution of the algorithm is in general around 4 for nucleic acids (this version runs 4 times faster) which is good but not enough. We therefore show in next section a second idea that may dramatically improve that performance.

5 Sketching the Solution Space

5.1 Main Idea

The second idea for improving the practical performance of the algorithm came from the following experiments. We had run the algorithm looking for the longest models present in all four of a set of randomly produced sequences each of length 100 defined over the $\{A, C, G, T\}$ alphabet with the following cover:

$$WCC = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (S, 1)\ \forall S \in \mathcal{P}^{+}(\Sigma),\ \mid S \mid \neq 1 \end{array} \right\}.$$

The algorithm took 875 seconds on a Silicon Graphics (R4000) workstation to produce an answer even after the first improvement (section 4) was implemented.

We then run the algorithm under the same conditions except that this time we had:

$$WCC = \left\{ \begin{array}{l} (\{X\}, \infty)\ \forall X \in \Sigma, \\ (\{A, C, G, T\}, 2^{|\Sigma|-1} - \mid \Sigma \mid) \\ (S, 0)\ \forall S \in \mathcal{P}^{+}(\Sigma),\ \mid S \mid \neq 1\ and\ S \neq \Sigma \end{array} \right\}$$

that is, the only sets permitted in the models were the unitary sets and the wild card, this last set being allowed at most 11 (= 15 - 4) times in the models. Furthermore, we looked this time for all models of length $k^{1}_{max} = 15$ which was the length of the longest models found in the previous case.

In terms of sets of occurrences, it is clear that we have: space solution(experi- ment 1) $\subseteq$ space solution(experiment 2).

The time needed for the algorithm to produce the solution to experiment 2 was 2 seconds on the same machine.

In terms of a search of the space of all models, experiment 2 can be seen as performing a coarse grained exploration of that space, where many branches of the tree are put together and examined in one single pass. What this simple experiment shows is that proceeding in such a way can be much quicker that trying to look for exact answers from the start. If that first exploration strictly retains the general contours of the solution space while still being able to eliminate a lot of candidate space, then in a second phase of the algorithm we need only explore the subspace delimited by the first phase. This approach may be very advantageous if the coarse grained search can be performed in a much faster way than the fine grained one and the solution space it delineates is much smaller than the initial search space, or if it is able not only to limit the space that needs to be further explored but can also put some order into it.

5.2 Algorithm

Given the cover WCC defined by:

$$WCC = \left\{ \begin{array}{l} (\{X\}, a)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, b)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, c)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, d) \end{array} \right\}$$

where a, b, c, d are non-negative integers and b, c, d are finite, the scheme to follow to solve either problem 1 or 2 of section 2 is then the one indicated in figure 2.

The small modification that must be done to the algorithm in the second pass (points 1.2 and 2.2) concerns just the first step, that is the construction of the sets $Occ(M)$ for $\mid M \mid = 1$ and $w_M \geq 1$. These sets are now obtained by sliding along sequence s as before but this time the elements (i, s_i) that are placed in $Occ(M)$ are those for which $s_i \in M$ <u>and</u> i is a multiple of k (since we are working in this pass with a concatenation of the occurrences).

Running this two-phases version of the algorithm on the same set of four randomly generated sequences of length 100 each looking for the longest models present in all four of them, we obtained the correct answer in 38 seconds, which represents a time factor of more than 25 gained over the previous version. The final version runs therefore a hundred times faster than the straighforward

1. Resolution of problem 1 - search of all models of length k satisfying the quorum q:
   1.1. do a first search with the cover WCC' defined by:

$$WCC' = \left\{ \begin{array}{l} (\{X\}, a)\ \forall X \in \Sigma = \{A, C, G, T\}, \\ (\{X, Y\}, 0)\ \forall X, Y \in \Sigma, \\ (\{X, Y, Z\}, 0)\ \forall X, Y, Z \in \Sigma, \\ (\{A, C, G, T\}, b + c + d) \end{array} \right\}$$

   1.2. for each model M found in the first pass
   concatenate its occurrences and run a slightly modified version
   of the algorithm (see section 5.2) this time with cover WCC
   and looking for all valid models of length k
2. Resolution of problem 2 - search of all models of greatest length satisfying the quorum:
   2.1. do a first search with cover WCC' given above
   let k'_{max} be the length of the longest models found
   2.2. for each model M found in the first pass
   concatenate its occurrences and run a slightly modified version
   of the algorithm (see section 5.2) this time with cover WCC
   and looking for all valid models of length k'_{max}
   2.3. if no valid model is found
   do {
   solve problem 1 for $k = k'_{max}$ - 1, then $k = k'_{max}$ - 2 etc
   }
   until at least one valid model is found
   /* note this last step can also be done in a dichotomic way */

Figure 2: An idea of the two-pass version of the algorithm.

approach. Further tests (results not given here) show that this gain is the bigger the greater is the number of sequences analized.

Let us observe also that working with a two-phases algorithm gives us an easy way for imposing a further constraint on the number of sets that may appear in valid models. This is obtained by fixing the weight attached to $\{\Sigma\}$ in WCC' not to $b+c+d$ but instead to a value noted f strictly smaller than $b+c+d$. In this case, models cannot contain more than f "special" sets, where a "special" set is any non unitary set specified in the second pass. This additional constraint proved to be useful in some practical cases.

5.3 Complexity

Again if quorum q is not taken into account and if $w_S = \infty$ for all $S \in \mathcal{P}^+(\Sigma)$, the time complexity of the first phase of the algorithm is majored by $O(n.N.k.2^k)$ and that of the second phase by $O(m.k.p)$ where m is the total number of occurrences found in the first phase. In general, m will be much smaller than $n.N.2^k$. Since the first and second phases of the algorithm have to be repeated at most k'_{max} times where k'_{max} is the length of the longest models found with cover WCC', the overall time complexity of the algorithm is bounded over by $O(n.N.(k'_{max})^2.2^{k'_{max}} + m.(k'_{max})^2.p)$.

As a matter of fact, the value of k_{max} for the cover WCC can be searched for in a dichotomic way from that of k'_{max} for the cover WCC' and the actual bound is therefore given by $O(n.N.k'_{max}.\log k'_{max}.2^{k'_{max}}.2^k + m.k'_{max}.\log k'_{max}.p)$. Where fixed k-length models are looked for, the bound becomes $O(n.N.k.2^k + m.k.p)$.

As with the straightforward approach, the introduction of a quorum constraint $q > 1$ and the fact that $w_S \neq \infty$ for most S in the cover mean that, in practice, the average behavior of the algorithm is much better than that.

6 Restricted Combinatorial Covers With Errors

As mentioned in the introduction, it is sometimes interesting to allow errors (substitutions, deletions and insertions) between a model and its occurrences. In two previous papers, this is what we did both for models defined as words over the same alphabet Σ as that of nucleotides [22] and for models defined as products of sets of a cover C of the alphabet Σ of amino acids [25].

Errors can be used with weighted combinatorial covers too. In the case of this paper, we first introduce them for restricted covers $RWCC$. These are covers for which all subsets except the wild card receive an infinite weight:

Definition 6.1 *A restricted weighted combinatorial cover $RWCC$ of Σ is defined by:*

$$RWCC = \left\{ \begin{array}{l} (S,\infty)\ \forall S \in C \ where\ C\ is\ a\ set\ of\ proper\ subsets\ of\ \Sigma, \\ (S,0)\ for\ S \in \mathcal{P}^+(\Sigma) \setminus C, \\ (\{\Sigma\}, w)\ where\ w\ is\ a\ non-negative\ finite\ integer \end{array} \right\}.$$

What we obtain is then an algorithm able to find all models M either of a fixed length k or of the greatest possible length k_{max} defined over a cover $RWCC$ together with their occurrences, where these occurrences consist now in all the words u in the sequences that present at most e errors with a nearest word $v \in M$.

The time complexity of this algorithm is bounded over by $O(n.N.k^{2e+1}.g^k.\ (\mid \Sigma \mid +1)^{e+1})$. This can be established in a similar way as in [25].

7 Examples of Applications

7.1 Preliminary Observations

The purpose of this section is to present some performance results in real biological situations and to suggest a few typical applications of the algorithm. As a matter of fact, the algorithm presented in this paper is essentially a "searching engine". It may therefore require further adaptations to specific biological problems or it should be embedded in a more "strategy oriented" program (for instance, a multiple alignement program) to be fully exploited. Examples of such adaptations together with suggestions about possible strategies are given below.

All the experiments were done on a Silicon Graphics Indy station (R4400) with 64Mb RAM. Performance results are given for a first prototypal, non optimized implementation of the algorithm.

7.2 Example of Application on Nucleic Acids

This first example serves to illustrate the "double combinatorial" aspect of the algorithm, that is, its ability to search for models while optimizing on the alphabet at the same time. Indeed, because of the small size of the nucleic acids alphabet, we can use a cover that corresponds to all the 15 non empty subsets of Σ. We test here the following three typical covers:

1. $WCC_1 = \left\{ \begin{array}{l} (\{X\},\infty)\ \forall X \in \Sigma = \{A,C,G,T\}, \\ (\{A,C,G,T\},0), \\ (S,1)\ for\ all\ other\ (10)\ sets \end{array} \right\};$

2. $WCC_2 = \left\{ \begin{array}{l} (\{X\},\infty)\ \forall X \in \Sigma = \{A,C,G,T\}, \\ (S,1)\ for\ all\ other\ (11)\ sets \end{array} \right\};$

3. $WCC_3 = \left\{ \begin{array}{l} (\{X\},\infty)\ \forall X \in \Sigma = \{A,C,G,T\}, \\ (\{A,G\},2)\ (at\ most\ 2\ purines), \\ (\{C,T\},2)\ (at\ most\ 2\ pyrimidines), \\ (S,0)\ for\ all\ other\ (9)\ sets \end{array} \right\}.$

Cover WCC_1			
N	t (in s.)	k_{max}	N_{mod}
10	5.2	9	405
50	45.0	8	143
100	62.2	8	2
140	70.2	8	1
Cover WCC_2			
N	t (in s.)	k_{max}	N_{mod}
10	1.2	10	3
50	119.0	8	12152
100	133.3	8	1966
140	152.2	8	1980
Cover WCC_3			
N	t (in s.)	k_{max}	N_{mod}
10	19.2	7	1
50	76.6	6	83
100	144.9	6	14
140	197.3	6	17

Table 1: Some performance results for fully combinatorial covers on the nucleotides alphabet.

In all the experiments, we work with the further constraint that the total number of "special" sets in solution models, that is non unitary sets (see section 5.2), is limited to 4. This avoids producing models that may be too degenerated.

We use as example promoter sequences of Bacillus subtilis from a recent compilation [9] recoverable through the World Wide Web (URL http://www.bio.cor nell.edu/microbio/helmann /helmann.htlm). The sequences are given in two groups, one supported by experimental transcript mapping (142 sequences), the other composed of putative promoters (94 sequences). We chose the first group as a test set.

The algorithm was run with the three covers given above, each time with a quorum set in percentage terms at 70%, with a varying number of sequences randomly chosen from the set of 142. In all cases, we looked for all valid models of maximum length. The CPU execution times (in seconds), maximum lengths and number N_{mod} of models obtained are given in table 1 for each number of selected sequences.

One of the more striking features of these results concerns the number of valid models obtained under certain conditions. Using cover WCC_2 for instance, with 140 sequences and a quorum of 70%, produces 1980 different valid models (for instance, one of them is: $TA[ACT]AA[AT][ACGT][AGT]$). The degree of non transitivity of the cover together with the quorum are two parameters that greatly affect this number but unfortunately they do so in a quite unpredictable way. Decreasing the quorum usually increases the number of models up to the point where longer (and thus fewer) models are found. Of course, most of these models appear to be variants of a same one. One of the reasons for the degeneracy observed in the models comes from the fact that several elements from the cover can be used at the most variable positions depending upon which set of occurrences corresponds to each model. When the quorum is not 100%, each of these particular sets may in this way lead to a slightly different version of a model. Although this results in numerous models, this kind of behavior can also bring an information that is useful when one is dealing with a non homogeneous set of

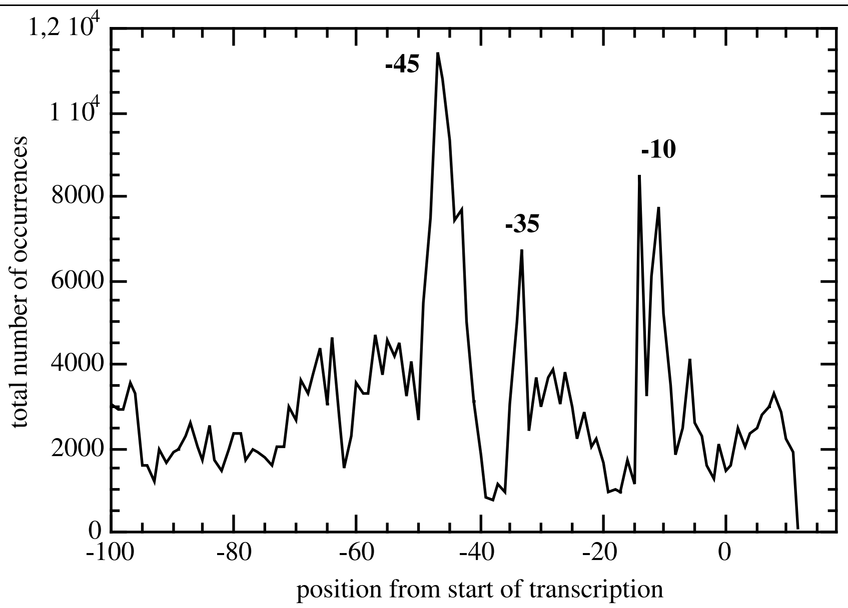

Figure 3: Plot of the number of occurrences of all 1980 longest models found with the cover WCC_2 given in the text, q =70% and for 140 promoter sequences. Position 0 of the X-axis indicates the known start of transcription.

sequences. In general though, the question comes of what to do with these models. The answer to this greatly depends upon the specific kind of application intended for the algorithm. We are now working on different ways to cluster models in order to give a synthetic (as opposed to exhaustive) list of results. The basic idea would consist in trying to build "metamodels", that is models of models. Another possible approach could be to statistically score each model separately and to sort the output accordingly.

This important issue of the degeneracy of the models can be more precisely illustrated in this case of the Bacillus subtilis promoter sequences since the expected signals are located in similar positions relatively to the known start of transcription. We plotted in figure 3 the number of occurrences of all 1980 models against their relative positions from this start (position 0). It is clear from this plot that most models occur on the two well-known positions -10 and -35 (two first peaks starting from the right side of the figure). However, another additional peak (actually the higher one) occurs before the -35 box (around position -45). Strict base conservation is not obvious in this region although it appears to be A-rich [9].

Closer examination of the models found (models are not presented here) shows that they are indeed mostly composed of As. Whether this peak results from the greater degeneracy of the models in that region or means that models are able to capture information, in particular about a possible positional correlation, that is ignored by simple base frequency/position analyses is still under investigation.

7.3 Example of Application on Proteins: Covers with the Wild Card and with Errors Allowed

Contrary to nucleic acids, it is not feasible for proteins to work with covers that correspond to all possible non empty subsets of the alphabet of amino acids as there are $2^{20} - 1$ of them. Moreover, most of these sets do not make any sense from a biological point of view. In the following experiment, we therefore chose one particular cover based on some physico-chemical and structural properties of the amino acids. This is the cover C given by:

$$C = \left\{ \begin{array}{ll} \{I, L, M, V\} & (hydrophobic), \\ \{A, G\} & (very\ small), \\ \{A, S, T\} & (small), \\ \{C\} & (cysteine), \\ \{F, Y, W\} & (aromatic), \\ \{K, H, R\} & (basic), \\ \{E, D\} & (acidic), \\ \{N, Q\} & (glutamine\ and\ glutamate), \\ \{G, P\} & (appear\ in\ constrained\ structures) \end{array} \right\}.$$

We then worked with the following weighted cover:

$$WCC = \left\{ \begin{array}{l} (S, \infty)\ \forall S \in C, \\ (S, 8)\ for\ \mathrm{S} =\ \Sigma, \\ (S, 0)\ for\ all\ other\ sets \end{array} \right\}.$$

As example, we used a set of 26 sequences from the XylS/AraC family of prokaryote transcriptional regulators described in a compilation [6] and extracted from SWISS-PROT release 31 (SWISS-PROT accession n° P19219, P05052, P11765, P03021, P07642, P03022, P17410, P25393, P10805, P26993, P23774, P28808, P10411, P28809, P09378, P09377, P27029, P16114, P22539, P29492, P28816, Q04248, P13225, P07859, Q05092 and Q05335). This family presents some nice features, the first one of which is a well-conserved pattern, referenced in PROSITE [1] as a "signature" pattern (PS00041), and known to be very specific of the family, together with several other more or less degenerated patterns which are located near the signature one, that is in the C-terminal part of the proteins (see [6] for more details). In particular, a very degenerated helix-turn-helix (HTH) pattern is thought to be located about 50 residues upstream from the signature.

The presence of such a "gradient" of patterns makes of this family a good test for checking the sensitivity of the algorithm. We used here the version that works with a more restricted cover and that allows for errors. We therefore run it with the cover indicated above and set the quorum at 100% (since in this case patterns are expected to be present in all the sequences). We also authorized one error (mismatch) between a model and any of its occurrences. Observe that because no set in the cover is strictly included in another one (except for the wild card), there is no "double combinatorial optimization" here as was the case with nucleic acids. The experiment yielded a total of 39 models of length 14 in about 6300 seconds of CPU time. Once again these models often correspond to variants of a same one and can actually be clustered into 5 different groups according to their similarities as shown in figure 4 where we also present the known PROSITE signature (initial line in bold and noted PR in the first column). The first group M1 found by our algorithm corresponds to this signature. The $[AG]/[GP]$ degeneracy at position 9 is simply explained by the fact that only G is in fact observed at this position and therefore all the sets that contain G are allowed. The models of this group fit quite well with the PROSITE pattern except at some positions (4, 8, 9 and 15) where the PROSITE pattern is more degenerated. One possible reason for this is that the PROSITE pattern was established from a larger set of sequences (38). However, examination of the occurrences of the models (not shown here) reveals that most of the time, the frequency of the symbols appearing in the PROSITE pattern and not in our models is low (for instance Q at position 9). With our approach, such "exceptions" are then captured in fact by allowing for errors in the occurrences as against the models.

Models M2, M3 and M4 correspond to other conserved parts occurring near the signature as mentioned in [6]. M2 actually splits in two subfamilies that are located around 20 residues downstream and 35 residues upstream from the signature. M3 and M4 appear to be located essentially 10 and 20 residues respectively upstream from the signature. Finally, we note that no model is found that corresponds to the HTH pattern. There can be two reasons for this. One may be because this pattern is too degenerated, the other because it is shorter than the first models found (let us recall that we asked for the longest possible models verifying a quorum of 100%). To check for this,

M1	
PR	**[LIV] XX[LIVMIA] [GSA]XXX [GNQ] [IFY]XXXXX [IF] X–XX[FY]XXXXXXP** X(4) [ILMV]XX[ILMV] [AST]XX[ILMV] [AG] [FYW]X(4) X(4) [ILMV]XX[ILMV] [AST]XX[ILMV] [GP] [FYW]X(4) X(1) [ILMV] [AST]XXX [AG] [FYW]XXXX[FYW] [FYW]X(1) X(1) [ILMV] [AST]XXX [GP] [FYW]XXXX[FYW] [FYW]X(1)
M2	X(2) [FYW] [KHR]XXX[AG]X[AST] [GP]XX[FYW]X(2) X(2) [FYW] [KHR]XXX[GP]X[AST] [GP]XX[FYW]X(2) X(1) [FYW]XXX[FYW] [KHR]XXX[AG]X[AST] [GP]X(1) X(1) [FYW]XXX[FYW] [KHR]XXX[GP]X[AST] [GP]X(1)
M3	X(2)X[ILMV]XX[AST]XXX[ILMV] [AST] [DE] [ILMV]X(2) X(1)X[ILMV]XX[AST]XXX[ILMV] [AST]X [ILMV] [AST]X(1) [ILMV]X[ILMV]XX[AST]XXX[ILMV]X X [ILMV] [AST]
M4	X(1) [ILMV]XXX[KHR] [ILMV]XX[AG] X[AST]X[ILMV]X(1) X(1) [ILMV]XXX[KHR] [ILMV]XX[AST]X[AC] X[ILMV]X(1) [FYW] [ILMV]XXX[KHR] [ILMV]XX[AG] XX X[ILMV]
M5	X(1) [ILMV]X[ILMV] [AST]X[ILMV] [AST]XXXXX[AST]X(1)

Figure 4: Longest models found in the XylS/AraC sequences with the cover WCC given in the text, $q = 100\%$ and one mismatch at most authorized between a model and any of its occurrences. X(i) means 0 to i X's.

we did the following (observe that this could suggest a more general strategy for manipulating the results of our algorithm): we ran again the algorithm on the left side of the strongest model found the first time and which corresponded to the PROSITE signature (a fully automated version of such a strategy would of course require that we run again the algorithm on the right side too), using the same parameters as before except that now the quorum was fixed to 70%. In this case, the HTH pattern is detected (in 6346 seconds of CPU time) among a total of 42 different models of length 15. Again, these models can be clustered into 5 families, the HTH one corresponding to 19 of the 42 models.

Recently, Neuwald *et al.* [18] have proposed a similar approach to constructing patterns using wild cards and a limited number and type of degenerated symbols (they concern only a few among all possible sets of two amino acids) and applied it to the same kind of examples. More precisely, their algorithm also performs a depth-first search of the space of patterns. Their method seems very sensitive and is faster than ours. This probably comes from the fact that the search is pruned using statistical rather than combinatorial criteria. At the present time though, we are still reluctant to introduce such statistical considerations at the level of the search since they do not guarantee exhaustiveness of the solution.

8 Conclusion

We introduced in this paper an algorithm that constructs all models identifying similar common features in a set of sequences while also exploring the alphabet of monomers in a combinatorial way. We use for this the concept of a weighted combinatorial cover. We showed algorithmic ideas that allow us to deal with such double combinatorics in a reasonably efficient way in most situations. One concerns a left-to-right minimality of the sets composing a model and the other a technique that involves making a sketch of the solution space before exactly exploring it. For some applications, the second idea in particular may greatly improve the performance of the algorithm.

We then described a version of this algorithm that works with restricted covers. These are covers for which all subsets except the wild card receive an infinite weight. In this latter case, errors are also permitted between a model and its occurrences.

We then showed a few preliminary biological applications of both versions with various variants on nucleic acid sequences and proteins that are encouraging and will be further explored in the future. We also mentioned persisting difficulties that are intrinsic to the nature of the problem and for which satisfying solutions still need to be proposed.

Acknowledgements

The authors would like to thank Professor Maxime Crochemore from the 'Institut Gaspard Monge' of the University of Marne la Vallée for very helpful discussions during the composition of this paper. His suggestions resulted in numerous improvements in the formalizations and general presentation.

The first author would like also to thank Professors Cyro de Carvalho Patarra, Siang Wun Song and Yoshiko Wakabayashi from the 'Instituto de Matemática e Estatística' of the University of São Paulo for their warm encouragements during all these years.

Last but not least, we thank Antoine Danchin for mentioning to us the helix-turn-helix problem and getting us interested in the subject, as well as Henri Soldano and Bernard Billoud for many discussions and exchange of ideas.

This work was supported by a grant from Organibio - Programme CM2AO.

References

[1] A. Bairoch. PROSITE: A dictionary of protein sites and patterns. *Nucl. Acids Res.*, 20:2013–2018, 1992.

[2] D. Bashford, C. Chothia, and A. M. Lesk. Determinants of a protein fold: unique features of the globin amino acid sequence. *J. Mol. Biol.*, 212:389–402, 1987.

[3] S. C. Chan, A. K. Wong, and D. K. Chiu. A survey of multiple sequence comparison methods. *Bull. Math. Biol.*, 54:563–598, 1992.

[4] B. Clift, D. Haussler, R. McConnell, T. D. Schneider, and G. D. Stormo. Sequence landscapes. *Nucleic Acids Res.*, 14:141–158, 1986.

[5] D. J. Galas, M. Eggert, and M. S. Waterman. Rigorous pattern-recognition methods for DNA sequences. Analysis of promoter sequences from Escherichia coli. *J. Mol. Biol.*, 186:117–128, 1985.

[6] M. T. Gallegos, C. Michan, and J. L. Ramos. The XylS/AraC family of regulators. *Nucl. Acids Res.*, 21:807–810, 1993.

[7] M. Gribskov, R. Luthy, and D. Eisenberg. Profile analysis. *Meth. Enzymol.*, 183:146–159, 1990.

[8] M. Gribskov, M. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Science USA*, 84:4355–4358, 1987.

[9] J. D. Helmann. Compilation and analysis of B*acillus subtilis* α-dependent promoter sequences: evidence for extended contact between RNA polymerase and upstream promoter DNA. *Nucleic Acids Res.*, 23:2351–2360, 1995.

[10] G. Z. Hertz, G. W. Hartzell, and G. D. Stormo. Identification of consensus patterns in unaligned DNA sequences known to be functionally related. *Comput. Appl. Biosci.*, 6:81–92, 1990.

[11] S. Karlin and G. Ghandour. The use of multiple alphabets in kappa-gene immunoglobulin DNA sequence comparisons. *The EMBO Journal*, 4:1217–1223, 1985.

[12] S. Karlin, M. Morris, G. Ghandour, and M.-Y. Leung. Efficient algorithms for molecular sequence analysis. *Proceedings of the National Academy of Science USA*, 85:841–845, 1988.

[13] A. Krogh, M. Brown, I. S. Mian, K. Sjoelander, and D. Haussler. Hidden Markov model in computational biology. Applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[14] A. M. Landraud, J. F. Avril, and P. Chretienne. An algorithm for finding a common structure shared by a family of strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(8):890–895, 1989.

[15] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wooton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262:208–214, 1993.

[16] C. E. Lawrence and A. A. Reilly. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins*, 7:41–51, 1990.

[17] H. M. Martinez. An efficient method for finding repeats in molecular sequences. *Nucleic Acids Res.*, 11:4629–4634, 1983.

[18] A. F. Neuwald and P. Green. Detecting patterns in protein sequences. *J. Mol. Biol.*, 239:698–712, 1994.

[19] J. Posfai, A.S. Bhagwat, G. Posfai, and R.J. Roberts. Prediction motifs derived from cytosine methyltransferases. *Nucl. Acids Res.*, 17:2421–2435, 1989.

[20] M. J. Rooman, J. Rodriguez, and S. J. Wodak. Relations between protein sequence and structure and their significance. *J. Mol. Biol.*, 213:337–350, 1990.

[21] M. J. Rooman and S. J. Wodak. Identification of predictive sequence motifs limited by protein structure database size. *Nature*, 335:45–49, 1988.

[22] M. F. Sagot, V. Escalier, A. Viari, and H. Soldano. Searching for repeated words in a text allowing for mismatches and gaps. Viñas del Mar, Chili, 1995. Second South American Workshop on String Processing.

[23] M. F. Sagot, A. Viari, J. Pothier, V. Escalier, and H. Soldano. Multiple comparison in biology: some mathematical formalizations of the problem and combinatorial approaches to solve it. submitted to *Discrete Applied Mathematics*.

[24] M. F. Sagot, A. Viari, and H. Soldano. A distance-based block searching algorithm. Cambridge, England, 1995. Third International Symposium on Intelligent Systems for Molecular Biology.

[25] M. F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. In *Proc. Combinatorial Pattern Matching Conf. 95*, volume 907 of *Lecture Notes in Computer Science*, pages 366–385, Helsinki, Finland, 1995. Springer-Verlag. to appear in *Theor. Comput. Science*.

[26] M. A. S. Saqi and M. J. E. Sternberg. Identification of sequence motifs from a set of proteins with related function. *Protein Eng.*, 7:165–171, 1994.

[27] G. D. Schuler, S. F. Altschul, and D. J. Lipman. A workbench for multiple alignment construction and analysis. *Proteins*, 9:180–190, 1991.

[28] R. P. Sheridan and R. Venkataraghavan. A systematic search for protein signature sequences. *Proteins*, 14:16–28, 1992.

[29] H. O. Smith, T. M. Annau, and S. Chandrasegaran. Finding sequence motifs in groups of functionally releated proteins. *Proceedings of the National Academy of Science USA*, 87:826–830, 1990.

[30] R. F. Smith and T. S. Smith. Automatic generation of primary sequence patterns from sets of related protein sequences. *Proceedings of the National Academy of Science USA*, 87:118–122, 1990.

[31] E. Sobel and H. M. Martinez. A multiple sequence alignment program. *Nucleic Acids Res.*, 14:363–374, 1986.

[32] G. D. Stormo. Consensus patterns in DNA. *Meth. Enzymol.*, 183:211–221, 1990.

[33] R. L. Tatusov, S. F. Altschul, and E. V. Koonin. Detection of conserved segments in proteins: Iterative scanning of sequence databases with alignment blocks. *Proceedings of the National Academy of Science USA*, 91:12091–12095, 1994.

[34] R. L. Tatusov and E. V. Koonin. A simple tool to search for sequence motifs that are conserved in Blast outputs. *Comput. Appl. Biosci.*, 10:0–0, 1994.

[35] W. R. Taylor. Pattern matching methods in protein sequence comparison and structure prediction. *Protein Eng.*, 2(2):77–86, 1988.

[36] W. R. Taylor. A template based method of pattern matching in protein sequences. *Prog. Biophys. Molec. Biol.*, 54:159–252, 1989.

[37] W. R. Taylor and D. T. Jones. Templates, consensus patterns and motifs. *Curr. Opin. Struct. Biol.*, 1:327–333, 1991.

[38] M. S. Waterman. General methods of sequence comparison. *Bull. Math. Biol.*, 46:473–500, 1984.

[39] M. S. Waterman. Multiple sequence alignments by consensus. *Nucleic Acids Res.*, 14:9095–9102, 1986.

[40] M. S. Waterman. Consensus patterns in sequences. In M. S. Waterman, editor, *Mathematical Methods for DNA Sequences*, pages 93–116. CRC Press, 1989.

[41] M. S. Waterman. Consensus methods for DNA and protein sequence alignment. In *Meth. Enzymol.*, volume 183, pages 221–237. 1990.

[42] M. S. Waterman, R. Arratia, and D. J. Galas. Pattern recognition in several sequences: consensus and alignment. *Bull. Math. Biol.*, 46:515–527, 1984.