# A distance-based block searching algorithm

M.-F. Sagot [1,3] A. Viari [1] H. Soldano [1,2]

[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] LIPN - Université de Paris Nord
URA CNRS 1507
Avenue J.B. Clément 93430 - Villetaneuse

[3] 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 the multiple comparison of a set of protein sequences. Our approach is that of peptide matching and consists in looking for all the words that occur approximatively in at least $q$ of the sequences in the set, where $q$ is a parameter. Words are compared by using a reference object called a model, that is itself a word over the alphabet of the amino acids, and the comparison between a model and a word is based on $w$-length words instead of single symbols. This idea is similar to the one used in the Blast program in the case of pairwise comparisons. Two $w$-length words are considered to be related if an alignment without gaps of the two using a similarity matrix has a score greater than a certain threshold value $t$. In our case, we say that a $k$-length word $u$ is an occurrence of a model $m$ of the same length if every $w$-length subword of $u$ is related to the corresponding subword of $m$ in the sense given above. If a model $m$ has occurrences in at least $q$ of the sequences of the set, $m$ is said to occur in the set. In percentage terms, the value of $q$ may correspond to something as small as 5% of the sequences (search for recurrent words in a set of non homologuous proteins) or as high as 70-100% (establishment of a list of all similar words as a first step in a multiple alignment program). The algorithm presented here is an efficient and exact way of looking for all the models, of a fixed length $k$ or of the greatest possible length $k_{max}$, that occur in a set of sequences. It can work with any kind of scoring matrix and an extension of the algorithm allows for the introduction of gaps between a model and its occurrences.

## 1 Introduction

The multiple comparison of sequences is an important but hard problem in molecular biology. The first difficulty comes from the fact that there is no clear manner of defining the concept itself. One natural way to proceed is to start from a well-established measure of pairwise comparison and to extend it to the case of a multiple one. For instance, the score of a multiple-sequence alignment can be defined as the sum of the scores of all projected pairwise alignments [4], or as the sum of the projected pairwise alignment score associated with each edge of an evolutionary tree [18] [19]. In each case, a function of similarity is defined, and an optimal alignment is one which minimizes this function. Dynamic programming is an algorithm that, in the special case of two sequences, provides an optimal solution in reasonable time and space. For more than two sequences, exhaustive algorithms become rapidly impracticable and various heuristics have been proposed [2] [6] [7] [22] [27].

Our interest is in comparing a big number of sequences simultaneously (typically a hundred or more) using a different approach. The method adopted is that of peptide matching or block searching, where a block is defined as an array of local sequence segments, not necessarily of equal lengths, that are similar in a certain way [8] [9] [13].

This method has been used by various authors previously with different definitions of similarity and for other purposes. Schuler's program [20], Macaw, is based on a SP (Sum of Pairs) approach to the scoring of a block of segments and has to resort to heuristics to reduce its search space of blocks. This program was later extended by Lawrence [12] through the use of a stochastic refinement method called Gibbs sampling. In a different approach, Soldano's paper [21] introduces a relation, reflexive, symmetric but in general not transitive between the amino acids and looks for maximal cliques of words. Sagot's paper [15] is an extension of that idea to the multiple comparison of protein 3D structures instead of sequences. However, both of these latter algorithms use symbols as units of comparison. Greater sensitivity would result if flexibility were introduced at the level of subwords instead of single symbols.

The purpose of this paper is therefore to use words as units of comparison. This is the main idea behind Blast's implementation [1] for doing pairwise comparisons between a query sequence and each one of the sequences in a database. Indeed, it establishes a relation of similarity between $w$-length words, that it calls $w$-mers, that is not a simple extension of a possible relation between symbols. Two $w$-mers are thus considered to be similar if an alignment without gaps of the two using a similarity matrix has a score superior to a certain threshold value.

The first phase of Blast consists in producing for each $w$-length word $u$ of the query a list of $w$-mers that are similar to $u$ in the sense given above. A second phase searches for all these $w$-mers in the database sequences. Any found constitutes what is called a hit which is further processed in a third and final phase.

To introduce a similar idea of a relation between $w$-mers in the multiple (as opposed to pairwise) comparison of sequences, we need to use an object called a model that is presented in [16] and [17] and that serves as a reference point against which the segments of a block are compared. So far as we know, the idea of using an external object as a reference point was first introduced by Waterman [24] [26] [25] who called these objects 'consensus patterns'. In Sagot's papers, models were defined either as words [16] or as subsets of the alphabet [17]. Here we define them as words as was done in [16]. This time however it is $w$-length words that are used as units in the comparison between a model and a segment.

Given a set of sequences to be compared, our algorithm works in two phases. In the first phase, we look for initial blocks of similarity of length $w$. This is done by finding all models of length $w$ for which there exist in at least $q$ of the sequences of the set $w$-length words that are similar to them (that is, that have with the $w$-length model a score superior to a certain threshold value $t$). The $w$-length model need never be exactly present in any of the sequences. We say of these models that they occur in the set and of the words similar to them that they are occurrences of the models. In the second phase, we try to extend the length of each block, that is, of each model found to occur in the first phase. This is done one unit at a time. The score of the last $w$-length subword of a model with its corresponding occurrences determines when the extension stops.

Although the algorithm can be used with nucleic acid sequences if an adequate scoring system is defined, our concern is mostly with protein sequences. The sequences we work with need not be homologuous. In that case, $q$ should be fixed to a small value, in percentage terms 5 or 10% (search for recurrent words [3]). On the other hand, if the sequences belong to the same family of proteins, $q$ should be fixed to a higher value, in percentage terms 70 or 80%, and the list of similar words produced by the algorithm can subsequently be used to construct a multiple alignment of the sequences [11] [23].

The time complexity is linear in the total length of the sequences, proportional to $k$ and to the maximum number of models found at any step $k' \leq k$ of the algorithm. This number can be considerably reduced if we impose that, for each model, each of its $w$-length subwords is exactly present at least once in the set. This gives a variant of the algorithm that is also described in this paper.

Section 2 defines the problem. Section 3 presents the algorithm and gives an idea of its time and space complexity. Section 4 introduces two variants. Section 5 presents a performance analysis on a real case and some results.

## 2 Statement of the problem

Let $\Sigma$ be the alphabet of the amino acids. The concept of model with which we work in this paper was introduced in [16]. We recall the definition of model it presented.

**Definition 1** *A model $m$ of length $k$ is an element of $\Sigma^k$.*

As we said, a model serves as a reference point against which the sequence segments are compared. This comparison is done in the following way.

Let a numerical matrix $M$ of size $\mid \Sigma \mid \times \mid \Sigma \mid$ be given such that:

$$M(a, b) = \text{score between } a \text{ and } b \text{ for all } a,\, b \in \Sigma.$$

If this score measures a similarity between $a$ and $b$, we talk of a similarity matrix (two well-known examples of which are PAM250 [5] and BLOSUM62 [10]), while if the score measures a dissimilarity between $a$ and $b$ we talk of a dissimilarity matrix. A special case of this latter matrix is when the dissimilarity measure is a metric, that is when the scores obey among other conditions the triangular inequality. In that situation, we talk of a distance matrix (an example of which is the matrix proposed by Risler [14]). The relation between a similarity and a dissimilarity matrix is simple and is given by:

$$M_D(a, b) = max(M_S) - M_S(a, b),\ \forall a, b \in \Sigma$$

where $M_D$ is the dissimilarity matrix, $M_S$ is the similarity matrix and $max(M_S)$ is its maximal value.

In what follows, when we talk of a matrix, we shall always be making reference to a similarity matrix.

**Definition 2** *Given $u = u_1u_2...u_k \in \Sigma^k$, $m = m_1m_2...m_k$ a model of length $k$ and $M$ a matrix, we note:*

$$score_M(u, m) = \sum_{i=1}^{k} M(u_i, m_i).$$

**Definition 3** *Given a numerical value $t$, a model $m$ and an integer $w$ between 1 and $\mid m \mid$, $m$ is said to be w-present in $s$ at location $i$ if the word $u$ of length $\mid m \mid$ starting at position $i$ in $s$ is such that:*

$$score_M(u_j...u_{j+w-1}, m_j...m_{j+w-1}) \geq t$$

*for all $j \in \{1,...,\mid m \mid -w+1\}$, that is if each $w$-length subword of $u$ has a score greater than or equal to $t$ with the corresponding subword of $m$.*

*We call $u$ a w-image of $m$.*

*We say that $(i, u)$ is a w-occurrence of $m$ in $s$ and we note $K(m)$ the set of all $w$-occurrences of $m$ in $s$, that is,*

$$\begin{gathered} K(m) = \{(i,u) \mid \\ \forall j \in \{1, ..., \mid m \mid -w+1\}, \\ score_M(u_j...u_{j+w-1}, m_j...m_{j+w-1}) \geq t\}. \end{gathered}$$

Let $\Sigma = \{a, b, c\}$, $w = 3$ and $t = 6$. Let $M$ be the following matrix:

| | a | b | c |
|---|---|---|---|
| a | 3 | 1 | 0 |
| b | 1 | 2 | 1 |
| c | 0 | 1 | 3 |

Given the three sequences:

s1 = abcbbabbbacabacbbbab
s2 = cabacaacbaccabcaccaccc
s3 = bbbacaccababacabacaba

then the longest model that is $w$-present in all three of these sequences is cacacacc (at positions 9, 1 and 12 respectively).

Figure 1: An example of models and their occurrences.

*Finally, we note:*

$$l_w(u, m) = score_M(u_{|m|-w+1} ... u_{|m|}, m_{|m|-w+1} ... m_{|m|}).$$

*It is just the score of the last $w$-length subwords of $u$ and $m$.*

Note that, according to this definition, a model may never be present exactly in the set of its occurrences.

**Example 1** *See figure 1.*

Observe that in the case where the scores in matrix $M$ obey a metric, the relation that is established between a model and its $w$-images induces a relation between the images themselves because of the triangular inequality.

**Statement of the problem** *Given parameters $t$, $w$ and $q$, a matrix $M$ of similarity and a set $\{s1, s2, ..., sN\}$ of sequences, we propose to solve the following problems:*

1. *find all models, and their occurrences, of a given length $k$ that are $w$-present in at least $q$ of the sequences in the set;*
2. *find the greatest length $k_{max}$ for which there exists at least one model of length $k_{max}$ that is $w$-present in at least $q$ of the sequences in the set, and solve problem 1 above for $k = k_{max}$.*

**Example 2** *See figure 1 for a resolution of problem 2 for three sequences and for $q = 3$.*

Let us observe finally that looking for models $w$-present in at least $q$ of the sequences of the set $\{s1, s2, ..., sN\}$ can be done by searching for the models $w$-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.

# 3 Basic algorithm

## 3.1 General description

The algorithm works in two separate steps: an initiation step and an extension step. The initiation step consists in efficiently finding all $w$-length models $w$-present in at least $q$ of the sequences

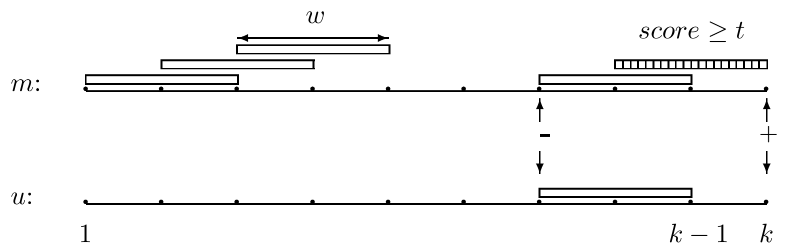


Figure 2: Extension of a model.

$s1, s2, ..., sN$. These models and their sets of occurrences are kept in an indexed table. The extension step picks each $w$-length model in turn, call $m$ this model, and tries to extend it as far as possible. This process stops when $m$ has occurrences in less than $q$ of the original sequences in the set (since in this case no longer model having $m$ as a prefix can itself have occurrences in at least $q$ of these sequences). This second step rests on a lemma that is given below.

## 3.2 Extension step

Let $s$ be a string of length $n$, $M$ a matrix, $m = m_1...m_k$ a model, $i \in \{1,...,n- \mid m \mid +1\}$ a position in $s$ and $u = u_1...u_k$ the $k$-length word starting at $i$ in $s$. Then the following lemma is verified:

**Lemma 1** *For all $(k-1) \geq w$:*

$$(i, u = u_1...u_k) \in K(m = m_1...m_k)$$

$$\Longleftrightarrow$$

$$(i, u_1...u_{k-1}) \in K(m_1...m_{k-1})$$
$$and\, l_w(u, m) \geq t.$$

This lemma gives a constructive property to build $K(m_1...m_{k-1}m_k)$ from $K(m = m_1...m_{k-1})$ for all $(k-1) \geq w$. We thus assume for now that all models of length $w$ that are $w$-present in $s$ have already been found and that we know their sets of occurrences. We discuss later how to compute this. Before we give the property, let us observe that the value of $l_w(u, m)$ can be calculated in $O(1)$ operations since we have:

$$l_w(u, m) = l_w(u_1...u_{k-1}, m_1...m_{k-1}) - \mathcal{M}(u_{k-w}, m_{k-w}) + \mathcal{M}(u_k, m_k)$$

knowing that:

$$l_w(u_1...u_{k-1}, m_1...m_{k-1}) = score_{\mathcal{M}}(u_1...u_{k-1}, m_1...m_{k-1})\ si\ (k-1) = w.$$

**Proposition 1** *Given a set of occurrences $K(m = m_1...m_{k-1})$ with $(k-1) \geq w$, we have:*

$$K(m = m_1...m_k)$$
$$=$$
$$\{(i, u_1...u_k) \mid (i, u_1...u_{k-1}) \in K(m_1...m_{k-1})$$
$$and\, l_w(u_1...u_{k-1}, m_1...m_{k-1}) - M(u_{k-w}, m_{k-w}) + M(u_k, m_k) \geq t\}.$$

Let $\Sigma = \{a, b, c, d, e\}$ and let $M$ be the following matrix:

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 5 | 3 | 0 | 2 | 1 |
| b | 3 | 4 | 1 | 2 | 4 |
| c | 0 | 1 | 5 | 4 | 2 |
| d | 2 | 2 | 4 | 4 | 1 |
| e | 1 | 4 | 2 | 1 | 4 |

Then the order matrix $O$ of $M$ is the matrix given by:

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| a | a | b | d | e | c |
| b | b | e | a | d | c |
| c | c | d | e | b | a |
| d | d | c | a | b | e |
| e | e | b | c | a | d |

Figure 3: An example of an order matrix.

Therefore, if for each element $(i, u)$ in $K(m)$, we keep the score of the last $w$-length subwords of $u$ and $m$, we can build the occurrences of any model $m$ knowing only the occurrences of its prefix of length $| m |$ - 1 (see figure 2). We shall see later that if the values of $M(a, b)$ are ordered for every $a \in \Sigma$ we can economise on quite a few operations.

### 3.3 Initiation step

We now face up to the problem we have been postponing, that is, to determine $K(m)$ for $m = w$. This need not be performed in an exhaustive way, that is, we do not have to verify for all models of length $w$ whether they have any occurrences in $s$. A simple observation allows us to restrict the search space at this step. This is that a word $u$ of $s$ cannot be a $w$-occurrence of a model $m$ if there exists $p \leq w$ such that:

$$score_M(u_1...u_p, m_1...m_p) + (w - p) * max(M) < t$$

where $max(M)$ is the maximal value of matrix $M$ as before.

In order to efficiently use this observation we must define an 'order matrix' which is related to $M$ in the following way:

**Notation 1** *Given a matrix $M$, we note $O$ the $| \ \Sigma | \times | \Sigma |$ order matrix of $M$ defined by:*

$$\forall\ a \in \Sigma, \forall\ j \in [1..(| \Sigma | - 1)],$$
$$M(a, O(a, j)) \leq M(a, O(a, j + 1)).$$

Each row of $O$ corresponds therefore to a symbol $a$ of $\Sigma$, and presents the symbols of $\Sigma$ ordered by decreasing score with $a$ (see figure 3).

### 3.4 Algorithm

The overall algorithm is given in figure 4.

```
/* Input */
        s = s1s2...sN: a string of length n composed of the concatenation of N sequences
        M: a matrix of similarity
        O: order matrix of M, as in notation 1
        t, w > 0, q: as indicated in the Statement of the problem
        k: a fixed length
/* Output */
        All models M of length k that have occurrences in at least q of the sequences
        s1, s2, ..., sN
/* Main data structures */
        K(m) = set of occurrences of a model m - contains for each occurrence not only
        its position i and the corresponding image u but also the value of l_w(u, m) noted
        more simply l_w: implemented as a stack
        ExtensionPossible = set of models that verify quorum of q and may therefore be
        extended: implemented as a stack
        T(m) = set of occurrences of model m of length w: implemented as a stack
        E = set of models of length w that have at least one occurrence in s

Main {
        for (i ∈ [1..(n − w + 1)])
                        /* note: λ is the empty word */
                Enumerate (i, i − 1, λ, 0, O, T, E);
        for (m ∈ E)
                if (T(m) ≠ ∅)
                        if (T(m) verifies quorum of q)
                                Explore (m, w, T(m));
}

/* Initiation step */
/* b and e are the beginning and end position respectively of the subword of length */
/* l in s, and m are the models in construction                                     */
Enumerate (b, e, m, l, O, T, E) {
        if (l = w)                                      /* end of recursion */
                if (score_M (s_b...s_e, m) ≥ t) {
                        T(m) = T(m) ∪ {(b, s_b...s_e);
                        E = E ∪ {m};
                }
        else {                                          /* body of recursion */
                if (l > 0 and score_M (s_b...s_e, m) + (max(M) * (w − l)) < t)
                        return;                         /* pruning */
                for (j ∈ [1.. | Σ |]) {
                        a = O(s_{e+1}, j);
                        Enumerate (b, e + 1, ma, l + 1, O, T, E);
                }
}
```

Figure 4: Algorithm for solving problem 1 - continues next page ...

```
/* Extension step */
Explore(m = m_1...m_l, l, K(m)) {
        if (l = k)                                  /* end of recursion */
                store m and K(m);
        else {                                      /* body of recursion */
                for ((i, u = s_i...s_{i+l-1}, l_w) ∈ K(m))
                        for (a ∈ Σ) {
                                l'_w = l_w - M(s_i, m_1) + M(s_{i+l}, a);
                                if (l'_w ≥ t) {
                                        K(ma) = K(ma) ∪ {(i, us_{i+l}, l'_w)};
                                        ExtensionPossible = ExtensionPossible ∪ {ma};
                                }
                        }
                for (ma ∈ ExtensionPossible)
                        if (K(ma) verifies quorum of q)
                                Explore (ma, l+1, K(ma));
        }
}
```

Figure 4: Algorithm for solving problem 1.

### 3.5 Time and space complexity

Given a string $s$ of length $n$, the initiation phase takes at worst $O(n. \mid \Sigma \mid^w)$ operations. This corresponds to the case where the matrix $M$ or the threshold $t$ are such that all possible $k$-length models are generated. On the other hand, with the identity matrix and with $t = w$, this step takes $O(n)$ operations.

Let us note $N_{mod}(q, k)$ the number of models of length $k \geq w$ $w$-present in $s$ at least $q$ times. Then it takes $O(n. \mid \Sigma \mid .N_{mod}(q, k))$ operations to build the models of length $k + 1$ from those of length $k$.

Now $N_{mod}(q, k)$ will depend on the string $s$, the matrix used, the choice made for the threshold $t$ and for $q$. It is still an open question how to calculate it. An important observation that we can do nevertheless, is that $N_{mod}(q, k)$ will not depend on $n$ and that the algorithm is therefore linear in $n$, although the contribution of $N_{mod}(k, q)$ to the time complexity can be great if $t$ is set to a very small number.

The space complexity is $O(n. \mid \Sigma \mid^w + n.k)$.

## 4 Variants of the algorithm

### 4.1 Variant 1

As we saw in the previous section, the time complexity depends on the maximal number of models of a given length that are $w$-present in s. That number can be dramatically reduced if besides requiring that a model $m$ be $w$-present at least $q$ times in $s$, we impose the further constraint that every $w$-length subword of $m$ be present exactly at least once. This means that models are built from actual sequence subwords.

Let us call $EK(m)$ the set of $w$-occurrences of $m$ in the sense defined above. Then:

$$EK(m) = \begin{cases} K(m) & if\ \forall j \in \{1, ..., \mid m \mid -w+1\}, \\ & \exists u \in K(m)\ such\ that \\ & score_M(u_j...u_{j+|m|-w+1}, \\ & m_j...m_{j+|m|-w+1}) \geq t \\ & and\ u_j...u_{j+|m|-w+1} = \\ & m_j...m_{j+|m|-w+1}; \\ \emptyset & otherwise. \end{cases}$$

Note that $m$ itself may still not be present exactly in $s$, since the constraint affects only its $w$-length subwords.

### 4.2 Variant 2: introduction of gaps

A second extension concerns the introduction of gaps (insertions or deletions) between the models and their occurrences. In our present implementation, the distance between two gaps cannot be smaller than $w$ and the score for aligning a gap with a symbol is a constant. That can be modified to allow for more closely placed gaps, and for non-constant gap penalties. The construction of a model allowing for gaps is done in the same iterative way as the construction without gaps. For each $w$-occurrence $(i, u)$ of a model $m$ a variable $g$ indicates the distance of the last gap present in the alignment of $m$ and $u$. When extending a model $m$ by one symbol, this variable is incremented and a gap is allowed between the extended model $m'$ and the corresponding occurrence $u'$ only if $l_w(u', m')$ $\geq t$ and $g \geq w$ ($u'$ is either the same as $u$ if the gap just introduced corresponds to a deletion, and to $u$ extended by two symbols if the gap is an insertion - in this last case, the inserted symbol is not taken into account when one considers the last subword of length $w$ of $u'$).

## 5 Performance

As mentioned in before, the algorithm running time strongly depends on the data, the similarity matrix used and the threshold value $t$. It is therefore important to evaluate the performance on a real case study rather than with artificial data. For this purpose, we extracted from SWISSPROT (Release 30) a set of 80 proteins belonging to the family of elongation factors. The members of this family are known to exhibit several well conserved regions (e.g. GTP binding sites). All the experiments were conducted on a Silicon Graphics (R4000) workstation. For the similarity matrix, we chose the widely used PAM250 matrix, except that we re-normalized its values in the [0, 100] integer range for convenience. Unless specified, the program parameters are:

- subword size $w = 3$
- score threshold $t = 210$ (note that the 3-length words scores range then from 0 to 300)
- minimum model length = 5
- quorum $q = 50\%$ or $70\%$

and we look for the longest possible models.

Using these parameters (and a quorum $q = 50\%$) on the 80 proteins, the longest models found are of length 10. A typical result is shown in figure 5, which displays the set of 46 occurrences of one of the longest models. Let us point out some of the differences between these occurrences. Some of them involve conservative replacements (e.g. R to K at position 4 or I to L or A at position 7) but some involve less favored replacements (e.g. G to P at position 1) as it could be expected from a word-based (as opposed to symbol-based) comparison.

To examine the behavior of the algorithm as a function of $n$, we ran the program with the same parameters as before (and a quorum of 70%) on an increasing number of sequences drawn randomly from the pool of 80 proteins. The result is shown in figure 6 (a), where the CPU time is plotted

| sequence | pos | occurrence | sequence | pos | occurrence |
|---|---|---|---|---|---|
| SUP2_YEAST | 383 | GQTREHALLA | STTN_RAT | 130 | GQTREHALLA |
| EFTU_YEAST | 149 | PQTREHLLLA | EFTU_THETH | 113 | PQTREHILLA |
| EFTU_THEMA | 113 | PQTREHVLLA | EFTU_SPIPL | 113 | PQTREHILLA |
| EFTU_SALTY | 112 | PQTREHILLG | EFTU_MYCHO | 113 | PQTREHILLA |
| EFTU_MYCGE | 113 | PQTREHILLA | EFTU_MYCGA | 113 | PQTREHILLA |
| EFTU_MICLU | 115 | AQTREHVLLA | EFTU_HALMA | 121 | PQTQEHVFLA |
| EFTU_FLESI | 113 | PQTREHILLA | EFTU_EUGGR | 113 | PQTKEHILLA |
| EFTU_ECOLI | 112 | PQTREHILLG | EFTU_CYAPA | 113 | PQTREHILLA |
| EFTU_CRYPH | 112 | PQTREHILLA | EFTU_CHLRE | 113 | PQTKEHILLA |
| EFTU_ASTLO | 113 | PQTKEHILLA | EFTU_ARATH | 180 | PQTKEHILLA |
| EFTU_ANANI | 113 | PQTREHILLA | EF1A_YEAST | 130 | GQTREHALLA |
| EF1A_STYLE | 130 | GQTREHALLA | EF1A_SOYBN | 130 | GQTREHALLS |
| EF1A_RHYAM | 79 | GQTREHALLA | EF1A_PLAFK | 130 | GQTKEHVLLA |
| EF1A_MOUSE | 130 | GQTREHALLA | EF1A_LYCES | 130 | GQTREHALLA |
| EF1A_EUGGR | 130 | GQTREHALLA | EF1A_DICDI | 133 | GQTREHALLA |
| EF1A_CRIGR | 130 | GQTREHALLA | EF1A_CANAL | 130 | GQTREHALLA |
| EF1A_ARTSA | 129 | GQTREHALLA | EF1A_ARATH | 130 | GQTREHALLA |
| EF1A_APIME | 130 | GQTREHALLA | EF1A_ABSGL | 130 | GQTREHALLA |
| EF13_XENLA | 130 | GQTREHALLA | EF13_RHIRA | 130 | GQTREHALLA |
| EF12_XENLA | 130 | GQTREHALLA | EF12_RHIRA | 130 | GQTREHALLA |
| EF12_DROME | 130 | GQTREHALLA | EF11_XENLA | 133 | GQTREHALLA |
| EF11_RHIRA | 130 | GQTREHALLA | EF11_HUMAN | 130 | GQTREHALLA |
| EF11_DROME | 130 | DQTREHALLA | EF10_XENLA | 130 | GQTREHALLA |

Figure 5: Occurrences of one of the longest models (length 10) found in a set of 80 sequences of elongation factors.

against the number $N$ of sequences. Since the length of the sequences in the set is quite homogeneous (with an average length of 540), the x-axis can be considered to be linear with the total number $n$ of amino-acids. As shown in the figure, the running time increases linearly with $n$. The high value obtained for $N = 5$ is due to the fact that, for such a small number of sequences, the models tend to be larger and hence, the running time is greater. This is shown in figure 6 (b) which displays the greatest length ($k_{max}$) of the models found under the same conditions. Note that, when $n$ increases, $k_{max}$ tends to remain constant (here $k_{max} = 7$). This is a typical behavior for a homogeneous set of sequences. If the proteins are not related, this length rapidly decreases to 0.

As mentioned before, the time complexity is strongly influenced by the parameters $t$ and $q$. The effect of $q$ is shown in figure 6 (c). In that case, we fix the number of sequences to 50 and let the quorum $q$ vary from 30% to 100%. The crosses correspond to the running time of the basic algorithm. As we can see, the running time rapidly increases at small quorum values. This comes from the fact that a greater number of longer models satisfy this quorum condition. Again, this behavior is typical of related proteins, unrelated ones allow much smaller values for $q$. As described in the variant 1 section, one possibility to reduce the search space is to require that each $w$-length subword of the models be present exactly at least once in the sequences. The effect of this variant is shown in the same figure (black dots). For most practical purposes, this additional constraint is not a very high price to pay considering the dramatic improvement one then obtains in some cases.

# 6 Perspectives

Defining a relation of similarity based on $w$-length words instead of single symbols yields a greater degree of fuziness in the comparison of protein sequences. This is one of the reasons for Blast's superior sensitivity compared to other database scanning programs. The algorithm presented in this paper uses the same idea to perform multiple comparisons, as opposed to pairwise ones. We are able at present to analyze up to a few hundred protein sequences simultaneously in a reasonable time. A future goal is to extend the algorithm so as to be able to use it for scanning a database from a multiple point of view.

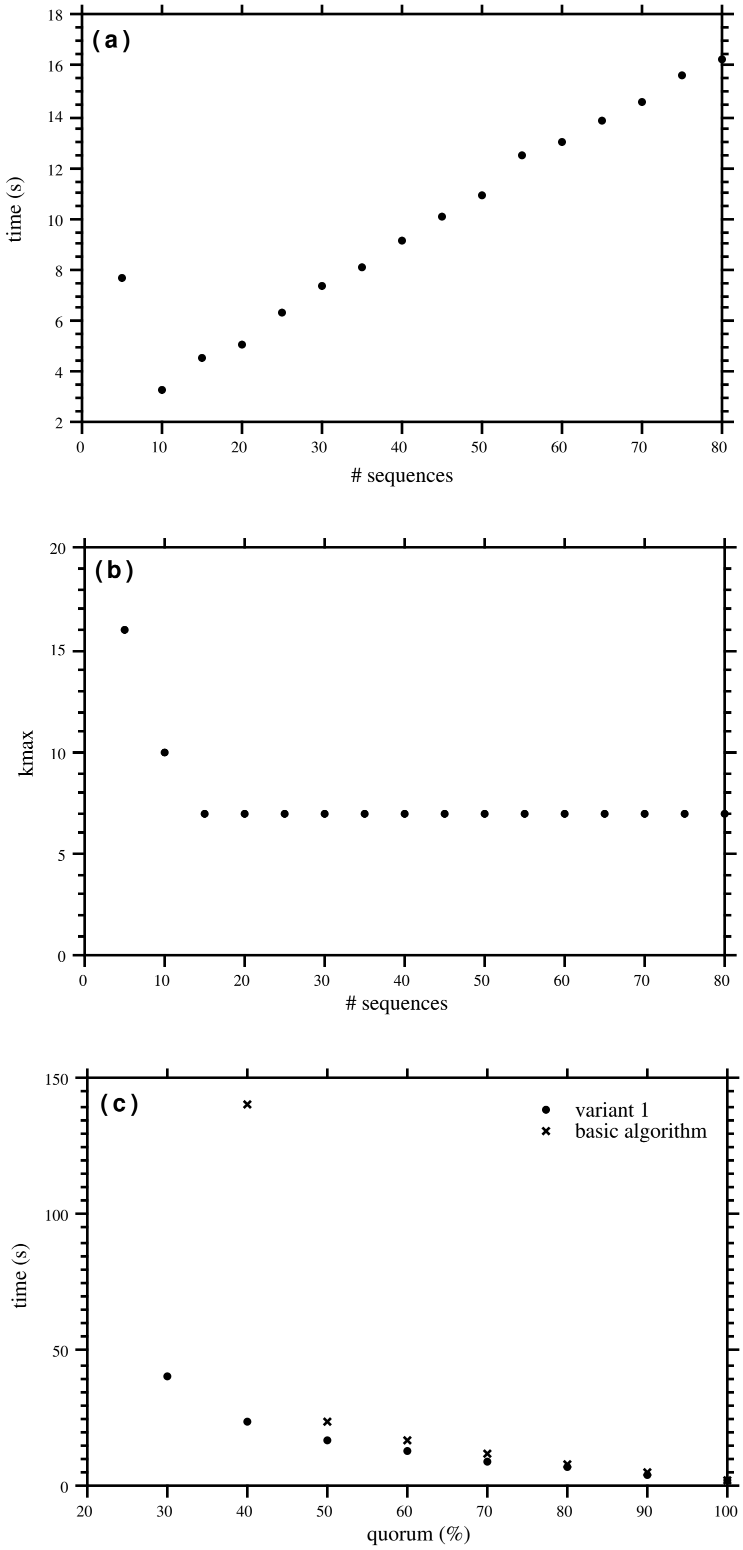


Figure 6: Performance tests of the algorithm (see text for the parameters). (a): CPU time against the number of sequences. (b): greatest length of the models found against the number of sequences. (c): CPU time against the quorum used and effect of variant 1.

# 7 Acknowledgement

The authors would like to thank Maxime Crochemore for very helpful discussions during the composition of this paper.

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

# References

[1] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.

[2] G.J. Barton and M.J.E. Sternberg. A strategy for the rapid multiple alignment of protein sequences: confidence levels from tertiary structure comparisons. *J. Mol. Biol.*, 198:327–337, 1987.

[3] B.E. Blaisdell, K.E. Rudd, A. Matin, and S. Karlin. Significant dispersed recurrent DNA sequences in the E*scherichia coli* genome. Several new groups. *J. Mol. Biol.*, 229:833–848, 1993.

[4] H. Carrillo and D.J. Lipman. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.*, 48:1073–1083, 1988.

[5] M.O. Dayhoff, R.M. Schwartz, and B.C. Orcutt. A model of evolutionary change in proteins. In M.O. Dayhoff, editor, *Atlas of Protein Sequence an Structure*, volume 5 suppl.3, pages 345–352. Natl. Biomed. Res. Found., 1978.

[6] D.F. Feng and R.F. Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, 25:351–360, 1987.

[7] D. Gusfield. Efficient method for multiple sequence alignment with guaranteed error bounds. *Bull. of Math. Biol.*, 55:141–154, 1993.

[8] S. Henikoff. Comparative sequence analysis: finding genes. In D.W. Smith, editor, *Biocomputing. Informatics and Genome Projects*, pages 87–117. Academic Press, 1994.

[9] S. Henikoff and J.G. Henikoff. Automated assembly of protein blocks for database searching. *Nucl. Acids Res.*, 19:6565–6572, 1991.

[10] S. Henikoff and J.G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA*, 89:10915–10919, 1992.

[11] A. Landraud, J.F. Avril, and P. Chretienne. An algorithm for finding a common structure shared by a family of strings. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11:890–895, 1988.

[12] 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.

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

[14] J.L. Risler, M.O. Delorme, H. Delacroix, and A. Hénaut. Amino acid substitutions in srtucturally related proteins: a pattern recognition approach. *J. Mol. Biol.*, 204:1019–1029, 1988.

[15] M. F. Sagot, A. Viari, J. Pothier, and H. Soldano. Finding flexible patterns in a text - an application to 3D molecular matching. *Comput. Application in the BIOSciences*, 11:59–70, 1994.

[16] 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.

[17] M.F. Sagot, A. Viari, and H. Soldano. Multiple comparison: a peptide matching approach. Helsinki, Finland, 1995. Combinatorial Pattern Matching Symposium. to appear.

[18] D. Sankoff. Minimum mutation trees of sequences. *SIAM J. Appl. Math.*, 28:35–42, 1975.

[19] D. Sankoff and R.J. Cedergren. Simultaneous comparison of three or more sequences related by a tree. In D. Sankoff and J.B. Kruskall, editors, *Time Warps, String Edits, and Macromolecules. The Theory and Practice of sequence Comparison*, pages 253–263. Addison-Wesley, 1983.

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

[21] H. Soldano, A. Viari, and M. Champesme. Searching for flexible repeated patterns using a non transitive similarity relation. *Pattern Recognition Letters*, 16(3), 1995.

[22] S. Subbiah and S.C. Harrison. A method for multiple sequence alignment with gaps. *J. Mol. Biol.*, 209:539–548, 1989.

[23] A. Viari and J. Pothier. *SmartMulti: a tool for the multiple alignment of protein sequences using flexible blocks.* Atelier de BioInformatique, 1994. in preparation.

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

[25] M.S. Waterman. Consensus patterns in sequences. In *Mathematical Methods for DNA sequences*, pages 93–114. CRC Press, 1989.

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

[27] A.K.C. Wong, S.C. Chan, and D.K.Y. Chiu. A multiple sequence comparison method. *Bull. Math. Biol.*, 55:465–486, 1993.