ELSEVIER

Computers and Chemistry 26 (2002) 421–436

Computers & Chemistry

www.elsevier.com/locate/compchem

Pairwise sequence alignment using a PROSITE pattern-derived similarity score

J.-P. Comet [a,*], J. Henry [b,1]

[a] *LaMI, Université d'Evry-Val d'Essonne, Cours Monseigneur Roméro, 91025 Evry Cedex, France*
[b] *INRIA Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex, France*

Received 30 April 2001; received in revised form 31 October 2001; accepted 12 November 2001

Abstract

Existing methods for alignments are based on edition costs computed additionally position by position, according to a fixed substitution matrix: a substitution always has the same weight regardless of the position. Nevertheless the biologist favours a similarity according to his knowledge of the structure or the function of the sequences considered. In the particular case of proteins, we present a method consisting in integrating other information, such as patterns of the PROSITE databank, in the classical dynamic programming algorithm. The method consists in making an alignment by dynamic programming taking a decision not only letter by letter as in the Smith & Waterman algorithm but also by giving a reward when aligning patterns. © 2002 Elsevier Science Ltd. All rights reserved.

Keywords: Dynamic programming; Sequence alignment; PROSITE; Databank scanning

1. Introduction

Sequence comparison has become a central notion in modern molecular biology. It aims at generating by induction information by comparing a new sequence to all sequences in annotated databanks. The pairwise alignment highlights the different zones of similarity. To evaluate a score of similarity, many methods are now available, allowing global alignments (Needleman and Wunsch, 1970) and gapped or ungapped local alignments (Smith and Waterman, 1981a,b). The Smith & Waterman algorithm uses the method of dynamic programming to give one of the best local gapped alignments. It leads to an alignment score that can be used as a basis for determining a possible homology. Most of the other approaches are based on heuristics (Altschul et al., 1990; Karlin and Altschul, 1990; Altschul et al., 1997).

All of these algorithms use a substitution matrix independent of the position. The similarity score is additive position by position with a fixed substitution matrix and a fixed function for gap penalties. But when the biologist makes an alignment without a computer, he favours some similarities depending on his knowledge of the structure and/or the function of the sequences. In the case of proteins he tries to put in correspondence patterns which are known to be pertinent for the considered sequences. To explore this direction, various extensions of the dynamic programming algorithm of Smith & Waterman have been developed in order to simulate a non-homogeneous

* Corresponding author. Tel.: +33-1-6087-3250; fax: +33-1-6087-3789.

E-mail addresses: comet@lami.univ-evry.fr (J.-P. Comet), jacques.henry@inria.fr (J. Henry).

[1] Tel.: +33-1-39635599.

substitution matrix. Wilbur and Lipman (Wilbur and Lipman, 1984) have presented a general framework for evaluating substitution depending on the context. The main reason for introducing the context dependence is that biological mutations appear in a non-uniform way along the sequences (Lewin, 1997). This implies a non-uniform distribution of the exact matches in the sequence alignments: there are regions where exact matches are more concentrated. On the other hand, if we align two random sequences which are independent and uniformly distributed, one observes that the exact matches are uniformly distributed (Huang, 1994). Then an alignment with five consecutive exact matches is more significant than another alignment with five non-consecutive exact matches. Huang also presents an algorithm taking into account the context favouring exact match if this appears in a well conserved region. Another idea has been developed by Barton (Barton and Sternberg, 1987) who used estimates of the secondary structure for each of the two sequences in order to limit the number of gaps inserted in regions of secondary structure.

In the context of multiple alignment several methods try to guide the multiple alignment process with the determination of conserved regions or patterns. The patterns do not come from a fixed library like PROSITE but rather are directly derived from the sequences to be aligned themselves, (see for example Smith and Smith, 1992; Posfai et al., 1994; Miller et al., 1994; Parida et al., 1998, 1999). If one wants to apply this idea to pairwise alignments, one has to focus directly on known functional patterns to be certain to take into account significant pairwise pattern alignments.

We present here an alignment method which takes into account the non-homogeneity of sequences and more precisely information linked to biological 'patterns'. Typically one thinks of protein sequences and functional patterns like those described in the databank PROSITE (Hofmann et al., 1999). One would like to favour the alignment of patterns. But in the case of an inversion process during evolution, two sequences can share two different patterns but in different order. Then it is impossible to find an alignment of sequences, preserving the order of letters, in which both pairs of patterns are matched. Then one cannot impose the alignment of all the patterns. The usual similarity score comparing the sequences letter by letter is modified to give a reward when patterns are matched. The new similarity score remains additive and the optimization problem can be solved by a dynamic programming approach.

Our method proceeds like the Smith & Waterman algorithm (Smith and Waterman, 1981b) by dynamic programming: subsequences increased letter by letter are compared and one attributes a supplementary bonus/reward when patterns are matched. The first step is to determine the occurrences of databank patterns in both sequences with a classical 'pattern-matching' algorithm. If one or more patterns are shared by both sequences, the second step implements a new dynamic programming algorithm for which the computation of the score has been modified: for each couple of position indices (i, j) which corresponds to the end of an alignment of two occurrences of the same pattern, it is also possible to align patterns weighting this new path.

In the first section we show that the Smith & Waterman algorithm does not always align patterns. Several behaviors are observed depending on the status of the occurrences. The second section focuses on the algorithm called SWP (Smith & Waterman algorithm with Patterns). We decompose the algorithm into two parts: the first one consists in aligning two occurrences of the same pattern respecting the matches in the motif, the second one is the general dynamic programming loop in which the objective function has been modified to take into account the possibility of aligning occurrences of patterns. The third section shows results of our algorithm in a large scale environment: the observed distribution of aligned occurrences respects our expectation. The larger the weight, the more occurrences are aligned. We tested also our algorithm in databank scanning, and observed that SWP makes clearer some relationships between sequences. This algorithm has been tested on all patterns from the databank PROSITE which are present in the protein databank Swissprot Rel. 35. When both sequences share several patterns the algorithm makes it possible to bind two similarity zones which are very far from each other.

2. Smith & Waterman alignments and Prosite patterns

2.1. Prosite patterns

PROSITE is a database of biologically significant protein sites and patterns formulated in such a way that with appropriate computational tools one can rapidly and reliably identify to which known family of proteins (if any) a new sequence belongs. The PROSITE functional patterns are represented by regular expressions. In this databank there are also other descriptions of biological functions (Matrix, Rules). In the PROSITE databank Rel. 14.0 one can find 1275 patterns, 56 matrices and four rules. But here we will only consider pattern data.

We eliminated some of the many patterns because they appear very frequently in the protein databank Swissprot Rel. 35. The non-informative patterns which have been eliminated are:

Pattern PROSITE	Occurrence number in Swissprot	Associated regular expression
PS00001	141147	N-P-[ST]-P
PS00004	42117	[RK](2)-x-[ST]
PS00005	332212	[ST]-x-[RK]
PS00006	377147	[ST]-x(2)-[DE]
PS00008	372978	G-{EDRKHPFYW}-x(2)-[STAGCN]-{P}
PS00009	23386	x-G-[RK]-[RK]
PS00013	7969	{DERK}(6)-[LIVMFWSTAG](2)-[LIVMFYSTAGCQ]-[AGS]-C
PS00016	4152	R-G-D
PS00029	4007	L-x(6)-L-x(6)-L-x(6)-L

Some of them are very short and often appear without the associated biological function. After elimination the databank consists of 1266 patterns.

To manipulate the PROSITE databank several tools are available. For example ProfileScan uses the method of Gribskov et al. (Gribskov et al., 1988) to find structural and sequence motifs in a protein sequence. ProSearch (Kolakowski et al., 1992) searches protein sequences for PROSITE database patterns. ScanProsite makes it possible to scan sequences from Swissprot or TrEMBL (Bairoch and Apweiler, 1997) for the occurrences of a particular pattern stored in the PROSITE databank.

Patterns are a certain characterization of biological properties shared by all sequences in a protein family. Nevertheless a pattern can appear in a protein without the associated biological property being expressed. Sequences which share the regular expression but not the biological property are called *false positive*, they are indexed in the PROSITE databank. In the same way some sequences share the biological property but do not present the regular expression. These sequences are called *false negative*, they generally are not indexed in the databank.

In the sequel, we will use the UNIX syntax for regular expression and the classical algorithm for regular expression recognition, regexp. To specify an amino acid one uses its representative character alone. The character '.' stands for any character, the logical 'or' is written '|', '[...]' stands for every character present inside the brackets and '[^...]' stands for every character not present inside the brackets. The symbols '^' and '$' represent the beginning and the end of the sequence. Finally '*', '+' have the classical meaning: '*' stands for any number of occurrences of the previous character including none and '+' stands for at least one occurrence of the previous character.

For example, the regular expression '^[^A][BC]A*B', describes strings which are at the beginning of the line, of which the first letter is not 'A' followed by 'B' or 'C' and by a certain number of 'A' and ending with a 'B' followed by another letter.

2.2. Does the Smith & Waterman algorithm align patterns?

Since we are looking for sequence alignments taking into account the information given by PROSITE patterns, we naturally show first that the classical Smith & Waterman alignment does not always match patterns shared by both sequences.

In order to test this fact, one has to distinguish five types of sequences sharing a particular pattern. All five of these types are annotated in the DR line of each entry of PROSITE:

1. The *true positives* (annotated by T) are sequences presenting the regular expression modeling the pattern and which belong also to the biological family associated with the pattern.
2. The *false negatives* (annotated by N) are sequences which belong to the family under consideration, but which do not contain the regular expression. Sequences which do not belong to the set under consideration and do not contain the regular expression, will be called true negatives.
3. The *false positives* (annotated by F) are sequences sharing the regular expression but which do not belong to the family.
4. The *potential* sequences (annotated by P) are the false negatives which do not contain the regular expression because the region(s) that are used as a 'fingerprint' (pattern or profile) is not yet available in the data bank (partial sequence).
5. Finally the *unknowns* (annotated by '?') are sequences for which one does not know whether they belong to the family or not.

For each type which contains the regular expression, we present the statistical behavior of the Smith & Waterman algorithm.

2.2.1. True positives

We want to know whether the classical dynamic programming algorithm does or does not align true positives. Fig. 1 shows the distribution of patterns from databank PROSITE Rel. 14 according to the number of non-aligned patterns divided by the total number of pairwise comparisons. In other words for each pattern of the databank,

- one computes the Smith & Waterman alignment for all pairs of true positives,
- one tests whether the Smith & Waterman alignment puts both occurrences of the regular expression in front of each other,

- one computes the number of alignments which do not align both patterns,
- finally the index r is the ratio of this number to the number of computed alignments.

The closer the index r is to 0, the more numerous are the aligned occurrences of the regular expression. If $r = 0$, then for all pairs of true positives, the classical dynamic programming does align the occurrences of the pattern. At the opposite if $r = 1$, for all pairs of sequences sharing the regular expression, the Smith & Waterman algorithm does not align instances of the pattern.

Generally the Smith & Waterman algorithm aligns occurrences of the same pattern when occurrences are close to the region of the highest similarity. The algorithm does not align them when occurrences are not similar or when they are very short and do not belong to the most similar region.

One can note that for many patterns the classical dynamic programming allows to highlight a region including the pattern. There are 556 patterns for which the classical dynamic programming aligns the pattern in all cases. These patterns are not very informative because the alignment regroups naturally these regions. In fact, either the pattern is very restrictive (the set of strings described by the regular expression is reduced to one element) or the occurrence of the pattern appears always in a region of high similarity. In the later case, one may ask whether the pattern is well defined. Since

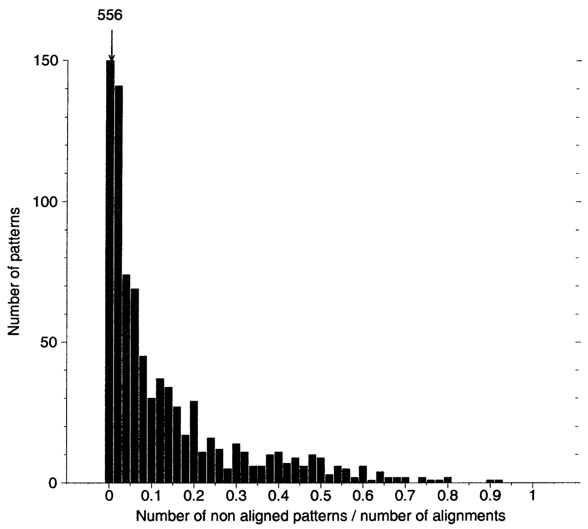


Fig. 1. Distribution of patterns from PROSITE Rel. 14.0 (true positives) according to the number of patterns which are not aligned by dynamic programming. The databank is Swissprot Rel. 35. For the first coordinate value 0, one observes 556 patterns. For these 556 motifs, the classical dynamic programming aligns by default both occurrences of patterns (both sequences are true positives).

in all cases the region of high similarity is larger than the regular expression, it may be possible to extend the definition of the regular expression.

In some other cases the dynamic programming does not align the pattern. However, for all true positives of Swissprot Rel. 35, the dynamic programming does align at least in one case the occurrences of the regular expression. In other words one cannot find one single pattern for which the index r is equal to 1. For a value of r close to 1 the histogram of Fig. 1 has some classes absolutely empty.

2.2.2. False positives

2.2.2.1. False positives versus false positives. In the same way, we want to know if the Smith & Waterman algorithm tends to align the same pattern in the case of two false positives. Generally it does not align patterns. Fig. 2 shows the distribution of patterns.

2.2.2.2. False positives versus true positives. In this case the results are similar to the case of two false positives (Fig. 2).

In such cases even if the biological function is not shared, the dynamic programming algorithm does align some patterns (Fig. 2, $r = 0$). This peak is much higher in the plot concerning two false positives than in that concerning a false positive versus a true one. One can align two false positive occurrences of the same pattern when they occur in the most similar zone between sequences. For example for the pattern PS00881 the Smith & Waterman algorithm aligns false positive occurrences in TIE1_BOVIN, TIE1_HUMAN and TIE1_MOUSE. All three of these sequences belong to the same family and the pattern occurs in the characteristic region of this family. This phenomenon does not happen in the case of aligning a false positive versus a true one. These homologies of two false positives could result from common biological properties different from the one related to the PROSITE motif.

3. The Smith & Waterman algorithm with pattern: SWP

Since the Smith & Waterman algorithm does not always align occurrences of the patterns, the alignment of these motifs can be a guide for an improvement of the sequence alignment. A better match between the occurrences is sought. This is an intermediate solution between the Smith & Waterman algorithm and the method used by biologists which consists in forcing the alignment of occurrences of motifs.

If there is only one pattern shared by both sequences, or if one seeks to align only one pattern, it is possible to impose the alignment of the motif and then to extend

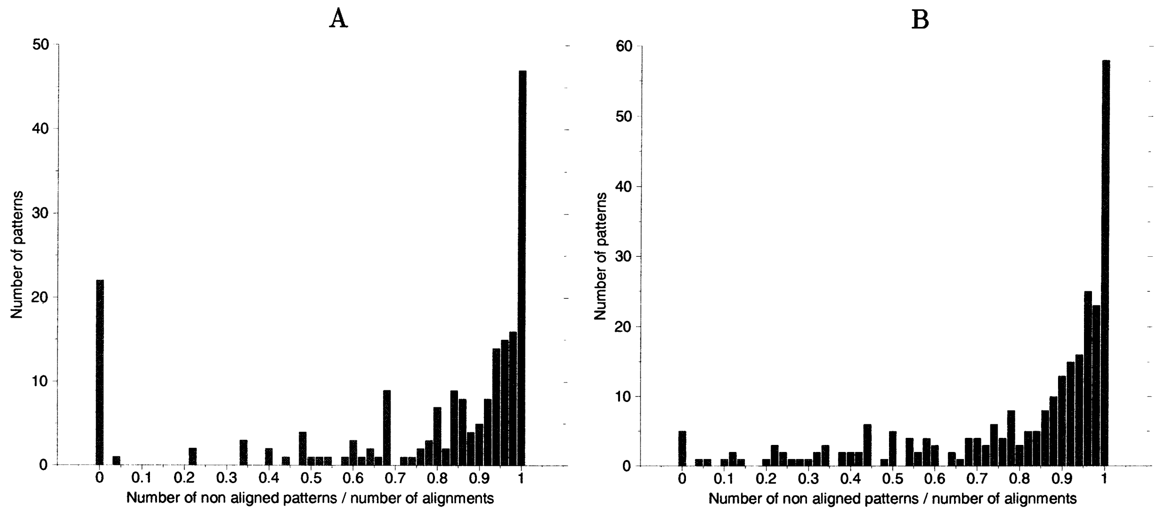
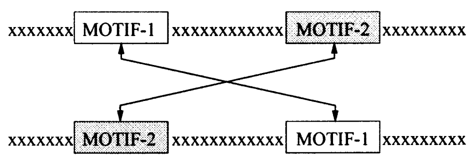


Fig. 2. Distribution of patterns from PROSITE Rel. 14.0 (false positives) according to the number of unaligned motifs. (A) False positive versus false positive. (B) False positive versus true positive.

the alignment on both sides with the Smith & Waterman algorithm.

However, if both sequences share several motifs which appear in different order, one can no longer force the alignment of the patterns. Let us consider that both sequences share two motifs in a different order:


xxxxxxx MOTIF-1 xxxxxxxxxxxx MOTIF-2 xxxxxxxxx
xxxxxxx MOTIF-2 xxxxxxxxxxxx MOTIF-1 xxxxxxxxx

If a biological relationship between these sequences exists, one is confronted with an inversion process. Classical algorithms are not able to align both motifs simultaneously, and one has to use algorithms which deal with inversion processes (Holloway and Cull, 1994). Note that our algorithm does not solve this inversion problem but chooses which pattern will be aligned.

The first stage is to determine all patterns which are shared by both sequences. Different programs make it possible to find in a sequence all occurrences of a set of patterns. After having two lists of the patterns present in both sequences, an intersection of the two lists gives the result. Another solution consists in taking advantage of the list given in PROSITE of all occurrences of a pattern in the databank Swissprot.

3.1. Alignment of two instances of the same pattern

The problem comes from the fact that a pattern can contain insertions/deletions. For example, if the regular expression contains the term '$x\{6, 8\}$', there is then at this place a string of which the length is between 6 and 8. Two instances of this pattern can have two different lengths. In that case the alignment has to contain an insertion/deletion in the zone defined by the term '$x\{6, 8\}$'.

On the other hand if the motif has a constant length, the alignment is very simple because there is no insertion/deletion. For example the pattern named ASP-Protease is defined by the following regular expression:

[LIVMFGAC]-[LIVMTADN]-[LIVFSA]-D-[ST]-G-[STAV]-[STAPDENQ]-x-[LIVMFSTNC]-x-[LIVMFGTA]

The sequences bar1_yeast and tryp_astfl share this motif. The pattern has two instances in bar1_yeast at the positions 60 and 284 and one in tryp_astfl at the position 194:

```
bar1_yeast:                    (V)(L)(F)D(T)G(S)(A)x(F)x(V)
                 60:   SQSLT    V  L  F D T G S  A D F W V    MDSSN

                               (V)(L)(L)D(S)G(T)(S)x(L)x(A)
                 284   TTKYP    V  L  L D S G T  S L L N A    PKVIA

tryp_astfl:                    (A)(A)(S)D(T)G(S)(T)x(L)x(G)
                 194   SGGPL    A  A  S D T G S  T Y L A G    IVSWG
```

Alignment of the first occurrence in bar1_yeast with the instance in tryp_astf1 will be:

```
 60    V L F D T G S A D F W V
             | | | |
194    A A S D T G S T Y L A G
```

If we decide to align the second instance in bar1_ yeast, one will obtain:

```
284    V L L D S G T S L L N A
             |   |       |
194    A A S D T G S T Y L A G
```

We define the score of the alignment as the sum of the substitution costs:

$$\text{Score}\ (\text{Word}_A,\ \text{Word}_B) = \sum_{k=0}^{\text{long}-1} S(A[\beta_A + k], B[\beta_B + k])$$

where

- Word$_A$ and Word$_B$ are the occurrences of the same pattern in sequence A and B, respectively;
- β_A and β_B are indices of the beginning of occurrences;
- long is the common length of both instances;
- $S(A[i], B[j])$ is the substitution cost between amino acids $A[i]$ and $B[j]$.

Nevertheless the alignment of occurrences of non-constant length patterns is tricky. For example the pattern Snake–Toxin is defined as:

CPx {6, 8} (L,I,V,Y,S,T)xCC

This pattern occurs in the sequences hcy_octdo and nx12_bunfl with two different lengths:

hcy_octdo:	CP x{6}	(Y)xCC
	CP SPEEPK	Y ACC
nx12_bunf1:	CP x{8}	(L)xCC
	CPEFTSRYKS	L LCC

So one cannot simply apply the Needleman & Wunsch algorithm since insertions/deletions have to occur in a well specified position. The Needleman & Wunsch algorithm does not guarantee that the insertion would take place at the right position. However, the alignment makes sense if one tries to align first the brackets and braces: (,), { and }. The idea is to align the sequences

{CP{SPEEPK}(Y)A(CC)}

and

{CP{EFTSRYKS}(L)L(CC)}

on an alphabet composed of 24 letters (20 letters for the amino acids and four for the brackets and braces). A new substitution matrix is then built on this new alphabet:

- the values of substitution of two amino acids do not change;
- perfect matches (‘{’, ‘{’), (‘}’, ‘}’), (‘(’, ‘(’) and (‘)’, ‘)’) are imposed (i.e. their edition cost is arbitrarily high);
- substitutions (‘{’, α), (‘(’, α), (‘}’, α), (‘)’, α) where α is an amino acid are impossible (value at $-\infty$);
- and substitutions (‘{’, ‘(’), (‘{’, ‘)’), (‘{’, ‘}’), (‘}’, ‘(’), (‘}’, ‘)’) and (‘(’, ‘)’) are also impossible.

The Needleman & Wunsch algorithm on this alphabet and with this substitution matrix gives a good alignment of patterns preserving insertion/deletion area. For the previous instances of the Snake–Toxin pattern, one has the following alignment:

```
{CP{--SPEEPK}(Y)A(CC)}           CP--SPEEPKYACC
||||        || | |||||  that is  ||          ||
{CP{EFTSRYKS}(L)L(CC)}           CPEFTSRYKSLLCC
```

The score of alignment is the score obtained by the previous method excluding the edition costs of brackets and braces.

A better solution

During the scanning of sequences for hypothetical pattern matching, one can seek possible positions of insertions/deletions. For example if the motif Snake–Toxin appears in the form CPSPEEPKYACC, the possible insertions/deletions introduced during the alignment with another occurrence of the same pattern, will take place in the region *SPEEPK* and only in this zone. When aligning two instances of the same pattern, there are two stages:

1. Search for possible positions of insertions/deletions. The following information is stocked:
   (a) the number of insertion/deletion areas;
   (b) for each insertion/deletion area, the indices (b_i, e_i) representing the beginning and the end.
   Two tables are built, one for the sequence A (indices (b_i^A, e_i^A)) and the other one for sequence B (indices (b_i^B, e_i^B)).
2. Alignment of occurrences of the pattern: let (i, j) be the current indices of alignment.
   (a) If (i, j) does not belong to any insertion/deletion area, the letter A_i faces the letter B_j in the alignment and the indices are set to $(i+1, j+1)$.
   (b) If (i, j) belongs to the insertion/deletion area number k, the Needleman & Wunsch algorithm is used on subsequences $A[\mathrm{b}_k^A,\mathrm{e}_k^A]$ and $B[\mathrm{b}_k^B,\mathrm{e}_k^B]$, and one goes to (a) with the indices ($\mathrm{e}_k^A + 1$, $\mathrm{e}_k^B + 1$).

This second solution is actually implemented.

3.2. Local alignment with pattern

After having defined an alignment for two occurrences of the same pattern, we present an algorithm for aligning sequences that takes into account the different patterns shared by them. For simplicity we consider here only linear penalty functions for gaps (gap-open penalty and gap-extend penalty are considered to be equal to δ). In the case of an affine penalty function the modifications are similar.

The Smith & Waterman algorithm consists in computing for each pair of indices (i,j) the score $M(i,j)$ for aligning the beginnings of sequences A[1, ..., i] and B[1, ..., j]. This is done with the following recurrence:

$$M(i,j)=\max\begin{bmatrix} M(i-1,j)-\delta, \\ M(i-1,j-1)+S(A[i],B[j]), \\ M(i,j-1)-\delta, \\ 0 \end{bmatrix}$$

A first method to favor the alignment of motifs consists in giving an additive reward (bonus) to the score when motifs match. With this new definition of the score, the dynamic programming algorithm can be applied but now at each step the decision variable consists in the choice of substitution, insertion, deletion of letters or in the alignment of a pattern. Let us suppose now that sequence A contains a pattern at position $(i-\text{long}_1+1, i)$ and that sequence B contains the same pattern at the position $(j-\text{long}_2+1, j)$, where long_1, and long_2 are lengths of pattern instances in sequences A and B, respectively.

We consider another path besides substitution, insertion and deletion: this new path is just the alignment of the pattern. The recurrence is then:

$$M(i,j)=\max\begin{bmatrix} M(i-1,j)-\delta, \\ M(i-1,j-1)+S(A[i],B[j]), \\ M(i,j-1)-\delta, \\ 0, \\ M(i-\text{long}_1,j-\text{long}_2)+\text{Score}(\text{Word}_A,\text{Word}_B)+\text{Reward} \end{bmatrix} \tag{1}$$

where

- Score $(\text{Word}_A, \text{Word}_B)$ is the score of the pattern alignment.
- *Reward* is a positive value rewarding the pattern alignment.
- $M(k,l)$ is the best score obtained when aligning $A[1, ..., k]$ and $B[1 ..., l]$.

In this new recurrence the score at the position (i,j) depends on the three neighbors like in the classical dynamic programming algorithm and also on another cell which can be far away from the current position: the score at the position corresponding to the beginning of pattern alignment.

In this method the alignment of motifs is not imperative and incompatible situations as in Fig. 3 can be dealt with.

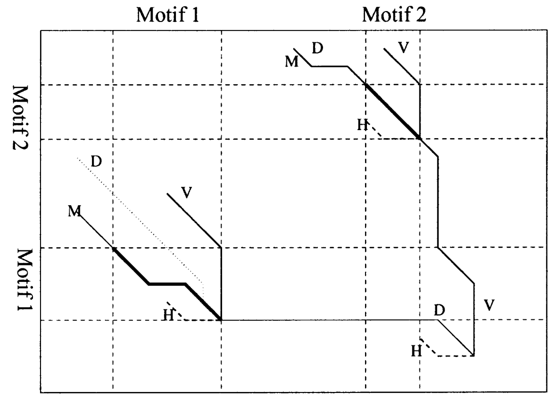


Fig. 3. Competition between patterns: If both sequences share two different patterns which are incompatible, the algorithm chooses which motif will be aligned according to the weight of each pattern. Lines with M correspond to the alignment of a motif, lines with H, D, or V to the insertion, substitution and deletion of the last character.

3.3. The basic recurrence of SWP

The SWP algorithm is a modification of Eq. (1) taking into account the case when several patterns end at the current position.

Let A and B be two sequences of length n and m, respectively. A couple of subsequences $(\text{word}_A, \text{word}_B)$ will be called a *pattern pair* if word_A and word_B belong to sequence A and B, respectively and if they are two occurrences of a same pattern.

Both sequences A and B can share several patterns. The same pattern can appear several times in the same sequence, and a position (i,j) can correspond to the end of several patterns. In this case the algorithm has to take the maximum among all alignments ending at the current position.

The general recurrence of the SWP algorithm is the following:

Table 1
Complexity of the SWP algorithm

	Complexity
Search for shared patterns	$O((n+m)\times t)$
Motif alignment	$O(l_{pA}+l_{pB}+\Sigma_{i=1}^{N}(\mathrm{e}_i^A-\mathrm{b}_i^A)\times(\mathrm{e}_i^B-\mathrm{b}_i^B))$
Dynamic programming	$O(m\times n)$

t stands for the motif databank size.

$$M(i,j)=\max\left(\begin{array}{c} M(i-1,j-1)+S(A[i\,],B[j\,]),\\ M(i-1,j\,)-\delta,\\ M(i,j-1)-\delta,\\ 0,\\ \max\left\{\begin{array}{c} p,\ \text{motif}\\ \text{ending at }(i,j\,)\end{array}\right\}\left(\begin{array}{c} M(i-\mathrm{long}_1^p,j-\mathrm{long}_2^p)+\\ \text{Score}\,(\text{Word}_A^p,\text{Word}_B^p)\\ \times\,\text{Reward}(p)\end{array}\right)\end{array}\right) \quad (2)$$

where

- Score(Word_A^p, Word_B^p) is the alignment score of two occurrences of the motif p,
- Reward (p) is a coefficient associated to the motif p weighting alignments of occurrences of this motif. Previously we introduced an additive weight (Eq. (1)) but here we use a multiplicative coefficient in order to try to weight according to the similarity of both instances of the motif. This multiplicative model will then favor the alignment of occurrences which are the most similar or the longest, that is which have the highest alignment score.

By this modification of the Smith & Waterman algorithm, not only the score is modified but as a result, the aligned subsequences. Throughout the sequel we make use of the multiplicative model with a reward coefficient independent of the motifs. However, shortcomings of this model will be discussed later.

3.4. Complexity

The algorithm is composed of three different steps: searching for all patterns shared by both sequences, aligning occurrences of patterns, and maximizing the score by the dynamic programming algorithm (Eq. (2)).

For two sequences of length m and n, respectively, the complexity of the pattern matching algorithm is linear in $O(m+n)$ because the search has to be done in both sequences. When all patterns from a databank are taken into account, the complexity of this stage increases proportionally to the size of the databank.

The complexity for the dynamic programming stage is quadratic that is in $O(m\times n)$ although a supplementary maximum operation has to be done for some cells.

For the alignment of the occurrences of a motif, two cases have to be considered:

- If the pattern has a fixed length, the algorithm is linear: the score is simply the sum of the substitution costs corresponding to each position of the alignment.
- On the other hand if the length of the pattern is variable, the computation of the score needs the knowledge of possible insertion/deletion areas. Then the dynamic programming is used for each insertion/deletion area. The complexity is:

$$O\left(\sum_{i=1}^{N}(\mathrm{e}_i^A-\mathrm{b}_i^A)\times(\mathrm{e}_i^B-\mathrm{b}_i^B)\right)+O(l_{p_A}+l_{p_B})$$

where N is the number of insertion/deletion areas defined by indices $(\mathrm{b}_i^A,\mathrm{e}_i^A)_{i\in[1,N]}$ and $(\mathrm{b}_i^B,\mathrm{e}_i^B)_{i\in[1,N]}$, l_{p_A} and l_{p_B}, are the lengths of occurrences of the pattern p in sequences A and B, respectively.

Table 1 sketches the complexity of the different stages of the algorithm SWP.

4. Discussion

4.1. Weight influence on alignments

Generally one expects the algorithm SWP to align more patterns than without weighting. Fig. 4 shows distributions of patterns according to two different weights, for true positive versus true positive or for false positive versus false positive. For true positives one observes the concentration of the distribution near zero. The higher the weight, the higher the peak of the distribution. For false positives the distribution also has a peak near zero (Fig. 4), but the proportion of non-aligned motifs is higher.

4.1.1. Sequences sharing only one pattern

When there is only one pattern shared by the sequences, the behavior of the SWP algorithm is simple. If the motif is not aligned by classical dynamic programming, then increasing the weight for the pattern tends to favor all paths which align occurrences of the pattern. An exception is when the occurrences are not similar to each other (see further).

Fig. 5 gives an example where the occurrences of the motif do not belong to the most similar region. When the weight is set to 2, the similarity area changes, the algorithm retains only the similarity region containing the pattern.

Generally if the patterns do not belong to the most similar region, increasing the weight for a pattern can produce an alignment including two similarity zones: one corresponding to the pattern alignment and the other one to the region highlighted by the Smith & Waterman algorithm. The reward can compensate for the gap cost necessary to connect the two distant similarity regions. For the previous example (Fig. 5) the gap cost is so important that the pattern weight cannot compensate for it to connect both similarity regions.

When there are several occurrences of the same pattern in both sequences, each corresponding to a distinct similarity region, tuning the weight can make it possible to align simultaneously these similarity zones in the same alignment. The maximum number of aligned patterns is also equal to the minimum of the two numbers of pattern occurrences in each of the sequences *A* and *B*.

4.1.2. Sequences sharing several patterns

As remarked before, the region containing the occurrence of a pattern can appear far away from a highly similar region. Let us suppose that there are two similarity zones very distant from each other. In other

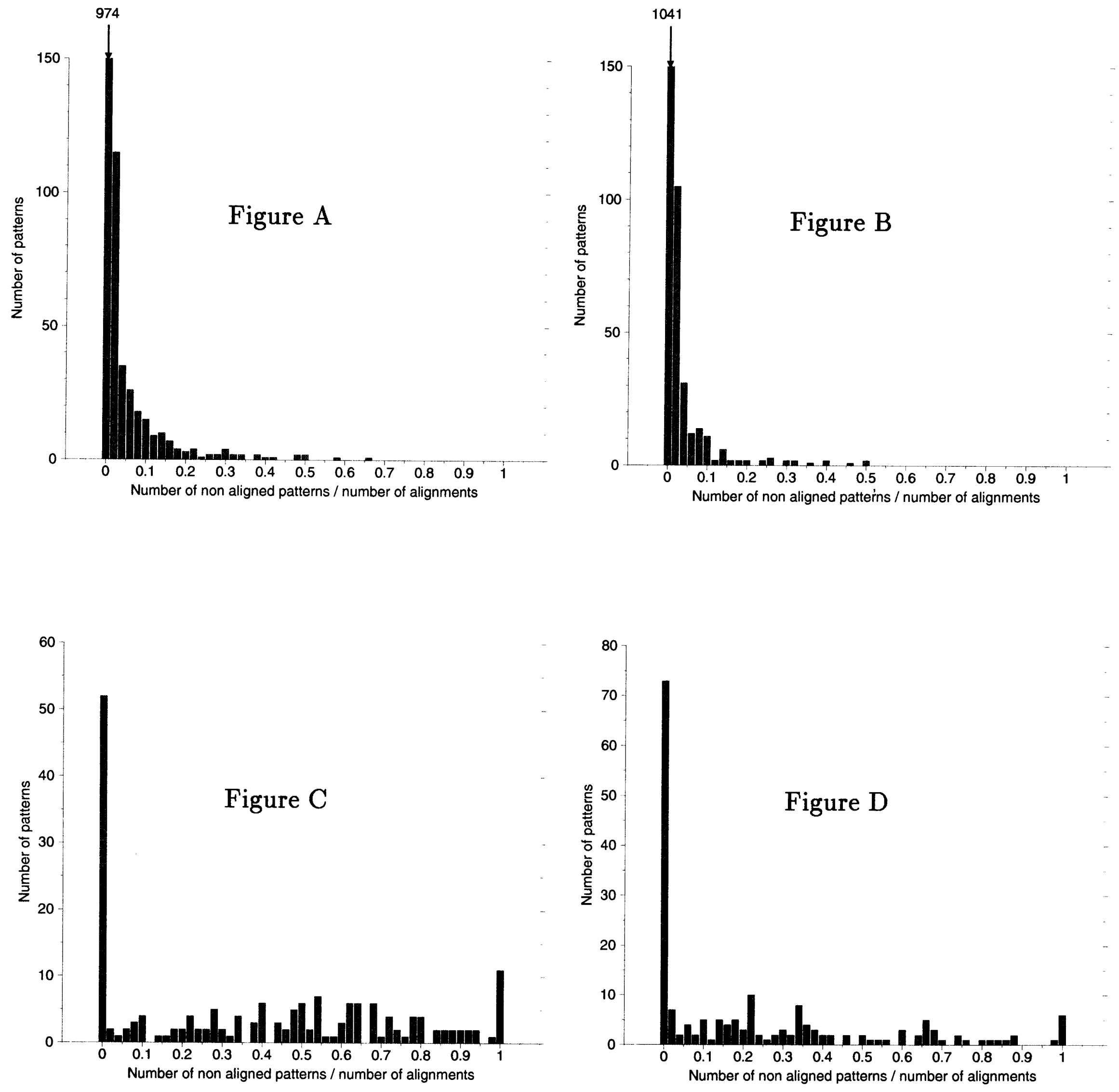


Fig. 4. Distribution of patterns form PROSITE Rel. 14.0 (true positive and false positive) according to the number of unaligned motifs. (A) (True positive): value for pattern weight = 2. (B) (True positive): value for pattern weight = 3. (C) (False positive): value for pattern weight = 2. (D) (False positive): value for pattern weight = 3.

```
Motif PS00841 : [VI][KRE]P.[FYIL]VFDG.\{2\}[PIL].[LVC]K

            Weight = 1

125 VTPEMAWKLIIALREHGIESIVAPYEADAQLVYLEKENIIDGIITEDSDM
    ||  |       |   ||  |||| || ||   |       | || ||| 
797 VTGQMCLESQELLQLFGIPYIVAPMEAEAQCAILDLTDQTSGTITDDSDI

175 LVFGAQTVLFKMDGFGNCITIRRNDIANAQDLNLRLPIEKLRHMAIFSGC
      ||| |  |   |           |      | |   ||   |   |  
847 WLFGARHV-YK-NFFSQNKHVEYYQYADIHN-QLGLDRSKLINLAYLLGS

225 DYTDGVAGMGLKTALRYLQKYP
    ||| |    |   |   |   |
894 DYTEGIPTVGYVSAMEILNEFP

{
            Score : 141

}

            Weight : 2

  1 GIKGLLGLLKPMQKSSHVEEFSGKTLGVDGYVWLHKAVFTCAHELAFNKE
    |  ||  ||             || | ||   ||   ||   |     |  
  1 GVQGLWKLLECSGRPINPGTLEGKILAVDISIWLNQAVKG-ARDRQGNAI

 51 TDKYLKYAIHQALMLQYYGVKPLIVFDGGPLPCKASTEQKRKERRQEAFE
        |    |    |      |  ||||     |  |  ||  |   |  
 50 QNAHLLTLFHRLCKLLFFRIRPIFVFDGEAPLLKRQTLAKRRQRTDKASN

101 LGKK
       |
100 DARK

{
            Score : 150
            SW Score : 119
            Motif alignment Score : 31
}
```

Fig. 5. Weight influences on alignments. For sequences XPG_XENLA and EXO1_SCHPO, the Smith & Waterman algorithm does not align the motif PS00841. Weighting the path corresponding to the pattern alignments makes it possible to highlight another similarity region with SW score (119 = 150-31) slightly lower than the SW score. The frame corresponds to the only aligned occurrences of pattern PS00841.

words a very long insertion/deletion is needed to vizualize in the same alignment these two regions. If a pattern is present in each of these regions, the SWP algorithm makes it possible to connect these regions in the same alignment.

Let us choose a motif from PROSITE for which the proportion of non-aligned occurrences among true positives is far from one.The motif *PS*01288 has four true positives in Swissprot Rel. 35: RTCB_ECOLI, Y682_METJA, YQ01_MYCTU and YT6J_CAEEL. Six alignments are possible. The Smith & Waterman algorithm does not align the occurrences of the pattern for three possible pairs (for example alignment between RTCB_ECOLI and Y682_METJA). The index *r* is then equal to 0.5. The sequences RTCB_ECOLI and Y682_METJA also share other patterns:

NAME	PROSITE	Regular expression
PKC_PHOSPHO_SITE	PS00005	[ST]-x-[RK]
CK2_PHOSPHO_SITE	PS00006	[ST]-x(2)-[DE]
MYRISTYL	PS00008	G-{EDRKHPFYW}-x(2)-[STAGCN]-P
UPF0027	PS01288	Q-[LIVM]-x-N-x-A-x-[LIVM]-P-x-I-x(6)-[LIVM]-P-D-x-H-x-G-x-G-x(2)-[IV]-G

Alignment without weight does not align pattern UPF0027 since the more similar region is far away from occurrences of this pattern. This region contains two patterns PS00006 and PS00008. The Smith & Waterman alignment is given in Fig. 6. Weighting pattern alignment leads to an alignment which contains clearly two similar zones separated by a long gap. The weight has been sufficient to compensate for the penalty due to the long gap.

The higher is the weight, the greater is the number of aligned motifs (Fig. 7). But how many occurrences can be aligned? Fig. 7 shows that at most 12 aligned patterns are found by our algorithm. In the following paragraph we compute the maximum number of occurrences which could be aligned.

4.2. Computing the maximum number of aligned motifs

The following table shows the number of occurrences of all patterns present in both sequences RTCB_ECOLI and Y682_METJA.

	RTCB_ECOLI	Y682_METJA	Symbol
PKC_PHOSPHO_SITE	8	11	P
CK2_PHOSPHO_SITE	2	6	C
MYRISTYL	11	13	M
UPF0027	1	1	V
Total	22	31	

Some of these occurrences overlap each other and the SWP algorithm is not able to align simultaneously such occurrences. On the other hand one has to consider also the order of patterns. If two motifs occur in both sequences in a different order, it is impossible to align them.

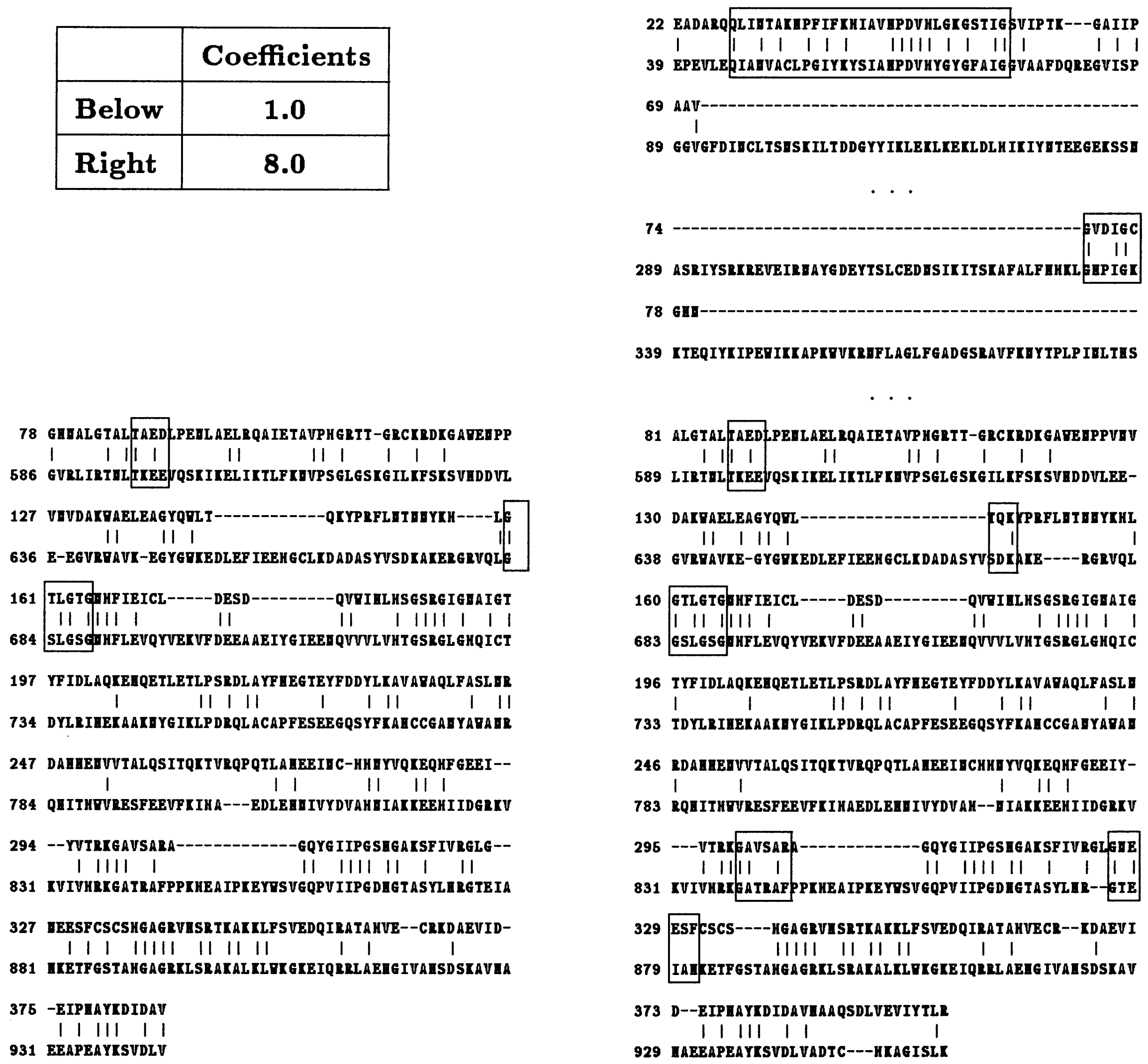
The following procedure gives a method to know the maximum number of motifs that can be aligned.

1. An alphabet with as many letters as different patterns shared by sequences is created. In the last example the alphabet is $\mathcal{A} = \{P, C, M, V\}$ (cf. previous table).
2. For each sequence one builds the series of letters corresponding to the orders of occurrences of different patterns. There can be several series of letters in case of overlap. In such a case only one pattern can be aligned. For the sequence Y682_METJA, one has two sequences:

   $\boxed{V}$ P P M P C P P M M C P P M P P M C C M P M M M C M M P

   $\boxed{M}$ P P M P C P P M M C P P M P P M C C M P M M M C M M P

   since motifs V and M overlap at the beginning of the sequence. These two sequences compose a first databank of motif series.A second databank is built for the second sequence RTCB_ECOLI. It contains 12 possible series of letters corresponding to the orders of occurrences of different patterns.
3. For each pair of sequences (A, B) where A belongs to the first databank and B belongs to the second, the longest common subsequence (with insertion and deletion) is computed. The result can be ob-



	Coefficients
Below	1.0
Right	8.0

Fig. 6. Weight influences on alignments (sequences RTCB_ECOLI and Y682_METJA). Increasing pattern weight makes it possible to introduce long gaps to connect distant similarity regions. Frames correspond to the pattern alignments.

tained by the Smith & Waterman algorithm using the following parameters:
- the substitution matrix is the identity matrix,
- gap-open and -extend penalties are set to 0.

4. The maximum score of all possible pairwise comparisons gives the maximum number of motifs which can be aligned. For the previous example there are $12 \times 2 = 24$ possible pairwise comparisons for which the maximum score is 15. Several alignments have this maximum score:

```
1 VPPMPCPPMMCPPMPPMCCMPMMMCMMP           3 PMPCPPMMCPPMPPMCCMPMMMCMMP
  |  | ||||| || | |  | |  |  |  S=15      || ||||| || ||      |||  |   S=15
1 V--M-CPPMM-PP-P-M--M-M--C--P           1 PM-CPPMM-PP-PP------MMC--P

1 VPPMPCPPMMCPPMPPMCCMPMMMCMMP           1 MPPMPCPPMMCPPMPPMCCMPMMMCMMP
  |  | ||||| || |     | |||  |  S=15      |  | ||||| || | |  | |  |  | S=15
1 V--M-CPPMM-PP-P-----P-MMC--P           1 M--M-CPPMM-PP-P-M--M-M--C--P

3 PMPCPPMMCPPMPPMCCMPMMMCMMP             1 MPPMPCPPMMCPPMPPMCCMPMMMCMMP
  || ||||| |  |     ||||| |    S=15      |  | ||||| || |     | |||  | S=15
1 PM-CPPMM-P--P-----PMMMC--P             1 M--M-CPPMM-PP-P-----P-MMC--P
```

For the sequences RTCB_ECOLI and Y682_METJA, at most 15 patterns can be simultaneously aligned. But we noticed before that our algorithm aligns at most 12 patterns. Why is our algorithm not able to align more than 12 patterns? The reason is that the multiplicative way of weighting is not well suited when the pattern alignment score is negative. We discuss this point in the following section.

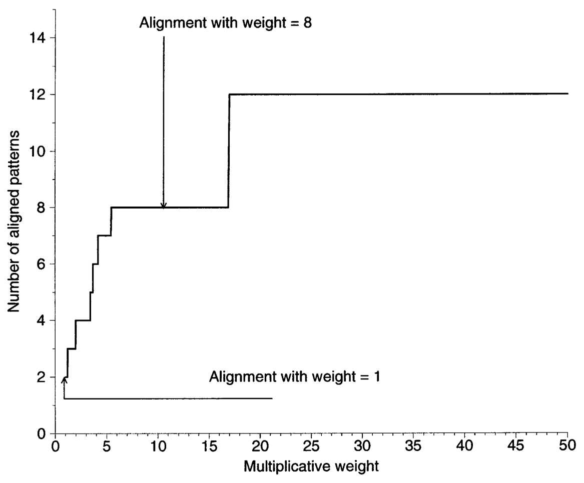


Fig. 7. Weight influence on the number of aligned occurrences of patterns (sequences RTCB_ECOLI and Y682_METJA). The higher the weight, the greater the number of aligned patterns. The saturation point seems to occur for 12 aligned motifs.

4.3. *Shortcomings of the multiplicative model*

When the regular expression is slack, that is when there are a lot of strings which match the regular expression, it may be possible to find two occurrences of the pattern for which the score is negative. In such case the multiplicative weight will never favor the alignment of occurrences. For example the pattern PS00011 is defined by the following regular expression:

.{12}E.{3}E.C.{6}[DEN].[LIVMFY].{9}[FYW]

The beginning of the pattern consists of 12 undefined letters. Biologically that means that the useful pattern 'E.{3}E.C.{6}[DEN].[LIVMFY].{9}[FYW]' cannot occur at fewer than 12 positions from the beginning of the sequence. Let us consider occurrences of the pattern PS00011 in OSTC_MACFA and FA10_BOVIN:

OSTC_MACFA: WLGAPAPYPDPLEPKREVCE-LNPDCDELADHIGFQEAY

FA10_BOVIN: FLEEVKQGNLERECLEEACSL-EEAREVFEDAEQTDEFW

With the blosum62 substitution matrix and the gap-open and gap-extend penalties equal to 10 and 1, the pattern alignment is:

```
       .{12}    E .{3} E.C  .{6} [E].[F]   .{9}   [W]
43 FLEEVKQGNLER E CLE  EAC SLEEAR  E V F EDAEQTDEF  W
    |           |      | |  |            |     |
 4 WLGAPAPYPDPL E PKR  EVC ELNPDC  D E L ADHIGFQEA  Y
       .{12}    E .{3} E.C  .{6} [E].[L]   .{9}   [Y]
```

and the alignment score is -1. This final score is the sum of the prefix alignment score ('FLEEVKQGN-LER' and 'WLGAPAPYPDPL') and of the alignment score of the *useful pattern*. The first one is -14 and the second is $+13$. The prefixes are at the origin of the negative score of the alignment: the prefixes do not bring any information to the motif. To favor this alignment one could implement an additive reward. But with the additive model two similar occurrences of a pattern would receive a reward equal to the one received by two distant occurrences. For example for the pattern '[DEN]. [LIVMFY]', the pattern alignment $\begin{bmatrix}\text{DLL}\\ \text{DML}\end{bmatrix}$ is probably more interesting than the alignment $\begin{bmatrix}\text{DLL}\\ \text{NGY}\end{bmatrix}$. The scores of these alignments with the blosum62 matrix are respectively, 12 and -4. With an additive model they receive the same reward, while with a multiplicative model the reward is proportional to the pattern alignment score and only the first one is favored.

4.4. Data-bank scanning with SWP

The problem consists in finding all known sequences from a databank which are similar to a query sequence. The different algorithms give a score which makes it possible to rank sequences from the most similar to the most distant. Is the SWP algorithm able to improve the ranking of the sequences? Is the information given by the pattern alignment sufficient to improve this databank ranking?

Since the SWP algorithm differs from the Smith & Waterman algorithm only for pairs of sequences which share at least one pattern, we present the results of databank scanning with a query sequence having a maximum number of patterns. If the query sequence does not contain any pattern, the SWP scores are strictly the same as those of the Smith & Waterman algorithm.

Table 2 gives the distribution of PROSITE patterns in Swissprot.

For example we choose the sequence FA12_HUMAN which has seven different motifs. We then compared all sequences from Swissprot which share at least one pattern with FA12_HUMAN: PS00021, PS00022, PS00023, PS00134, PS00135, PS01186 and PS01253. Table 3 shows for each pattern present in FA12_HUMAN, the number of sequences from Swissprot which share these patterns and also the total number of occurrences.

The patterns PS00022 and PS01186 have been put aside for complexity reasons since the number of occurrences was too large. Pattern PS00134 has not been retained because it was not so informative: [LIVM]-[ST]-A-[STAG]-H-C. Only patterns PS00021, PS00023, PS00135, and PS01253 have been retained. The databank of sequences sharing at least one of these four patterns has been created and contains 321 sequences.

Table 4 gives the 20 highest scores when comparing sequence FA12_HUMAN against one of the 321 considered sequences. Several coefficients for weighting have been tested. Globally one observed a stability of the order in which sequences appear. But with a weight greater than one the algorithm ranks some sequences at a higher position (see sequences FINC_HUMAN, FINC_BOVIN, FINC_RAT, and EL3B_HUMAN). For example, the sequence FINC_HUMAN occurs at the rank 273 with pattern weight equal to one and is ranked at position 87 with coefficient equal to three. Increasing the weight has made it possible to connect two similarity regions (Fig. 8).

The advantage of the SWP algorithm lies not only in the improvment of the ranking but also in the alignment itself. It makes possible in a large measure to correct the lack of sensibility of the Smith & Waterman algorithm, permitting the creation of long insertions/deletions.

4.5. Weighting problem

The main recurrence 2 uses coefficients to favor pattern alignment. The *Reward* (p) depends on the aligned pattern p since the information given by a pattern depends on its length and on its grammar. It remains hard to assign a numerical value to a pattern of the PROSITE databank.

Table 2
Distribution of PROSITE patterns in sequences from Swissprot

Number of PROSITE patterns (n)	Number of sequences from Swissprot Rel. 35 with n patterns
1	24 176
2	7610
3	2605
4	563
5	173
6	140
7	38
8	4

Table 3
Number of pattern occurrences present in FA12_HUMAN

PROSITE pattern	Number of sequences with pattern	Pattern occurrence number
PS00021	44	135
PS00022	321	1045
PS00023	29	55
PS00134	288	288
PS00135	287	287
PS01186	337	1071
PS01253	21	69

Table 4
Databank scanning with SWP (extract)

	Coefficient = 1 (SW)			Coefficient = 2			Coefficient = 3		
	Sequences	Scores	(1)	Sequences	Scores	(1)	Sequences	Scores	(1)
1	FA12_CAVPO	2507	4	FA12_CAVPO	3019	4	FA12_CAVPO	3531	4
2	FA12_BOVIN	2425	4	FA12_BOVIN	2935	4	FA12_BOVIN	3445	4
3	HGFA_HUMAN	1193	4	HGFA_HUMAN	1485	4	HGFA_HUMAN	1777	4
4	UROT_HUMAN	712	3	URT2_DESRO	852	3	URT2_DESRO	997	3
5	URT1_DESRO	708	2	URT1_DESRO	850	3	URT1_DESRO	993	3
6	URT2_DESRO	707	3	UROT_HUMAN	849	3	UROT_HUMAN	986	3
7	UROT_RAT	694	3	UROT_RAT	832	3	UROT_RAT	970	3
8	UROT_BOVIN	687	3	UROT_BOVIN	826	3	UROT_BOVIN	965	3
9	UROT_MOUSE	675	3	UROT_MOUSE	808	3	UROT_MOUSE	941	3
10	URTB_DESRO	666	2	URTB_DESRO	769	2	URTB_DESRO	872	2
11	UROK_BOVIN	663	2	UROK_BOVIN	763	2	UROK_BOVIN	863	2
12	UROK_RAT	656	2	UROK_HUMAN	754	2	UROK_HUMAN	854	2
13	UROK_HUMAN	654	2	UROK_RAT	750	2	UROK_RAT	844	2
14	UROK_PIG	639	2	UROK_PIG	739	2	UROK_PIG	839	2
15	UROK_PAPCY	628	2	UROK_PAPCY	728	2	UROK_PAPCY	828	2
16	UROK_MOUSE	623	2	UROK_MOUSE	718	2	UROK_MOUSE	813	2
17	PLMN_PIG	587	2	PLMN_PIG	687	2	PLMN_PIG	787	2
18	PLMN_BOVIN	582	2	PLMN_BOVIN	682	2	PLMN_BOVIN	782	2
19	PLMN_HUMAN	582	2	PLMN_HUMAN	682	2	PLMN_HUMAN	782	2
20	URTG_DESRO	571	2	URTG_DESRO	674	2	URTG_DESRO	777	2
87	CFAD_RAT	347	1	ENTK_MOUSE	412	1	**FINC_HUMAN**	475	2
88	TRY2_ANOGA	345	1	TRY2_ANOGA	412	1	**FINC_RAT**	474	2
100	EL2_BOVIN	334	1	TRYA_HUMAN	397	1	***FINC_BOVIN***	462	2
218	COGS_UCAPU	247	1	**FINC_HUMAN**	304	2	CTRB_GADMO	379	1
219	SNAK_DROME	247	1	**FINC_RAT**	301	2	TRYZ_DROME	377	1
224	NKP1_RAT	237	1	***FINC_BOVIN***	295	2	EL3B_HUMAN	372	1
273	**FINC_HUMAN**	147	1	TRYP_CHOFU	243	1	GRAB-MOUSE	305	1
274	***FINC_BOVIN***	147	1	MPRI_MOUSE	240	1	MCP1_PAPHA	303	1
279	**FINC_RAT**	142	1	GRL2_RAT	229	1	MCP2_MERUN	285	1

Sequence FA12_HUMAN has been compared with all sequences from Swissprot Rel. 35 which share at least one of the following patterns: PS00021, PS00023, PS00135 and PS01253. When the coefficient is one, the SWP coincides with the SW algorithm. Columns (1) give the number of aligned patterns.

The weight has to depend on the information level of the pattern. The more informative the pattern, the greater the associated weight. The information given by a pattern is inversely proportional to the probability of observing an occurrence of the pattern in a random sequence.

Information theory says that the information of an event is equal to $-\log_2(q)$ (McEliece, 1977) where q is the probability of the event. One has to compute or estimate the probability of observing a pattern in a random sequence. Nicodéme and co-workers (Nicodéme et al., 1999, 2002) presented a complete analysis of the statistics of the number of pattern occurrences in a random text. The characteristics of the distribution of the number of pattern occurrences are effectively computable, both exactly and asymptotically. These results would be a great help for evaluating the weight for each pattern from PROSITE.

5. Conclusion

We have proposed a new algorithm based on the dynamic programming approach which takes into account the information coming from the occurrences of PROSITE motifs. The similarity score of the Smith & Waterman algorithm has been modified to incorporate the presence of PROSITE patterns. The aim is to guide properly the alignment algorithm when patterns are present in both sequences. In databank scanning our algorithm SWP can reveal similarities between sequences which otherwise would be hidden. The rewarding method remains one possibility among others for improving the Smith & Waterman algorithm, allowing the creation of long insertions/deletions. For each pattern the weight has to be chosen. This point remains tricky, further work is needed to give a method for evaluating the weight according to the statistical

```
              Weight : 1

394 DQKYSFCTDHTVLVQTQGGNSNGALCHFPFLYNNHNYTDCTSEGRRDNMK
      ||     |||     |       ||||| |    |  ||  ||
27  EHKYKAE-EHTVVLTVTG-----EPCHFPFQYHRQLYHKCTHKGRPGPQP

444 WCGTTQNYDADQKFGFC
    || || | | ||  | |
71  WCATTPNFDQDQRWGYC

Motif 1 :

C.{2}PF.[FYWI].{7}C.{8,10}WC.{4}[DNSR][FYW]
.{3,5}[FYW].[FYWI]C


Motif 2 :

C.{6,8}[LFY].{5}[FYW].[RK].{8,10}C.C.{6,9}C


              Score : 147
```

```
              Weight : 3

394 DQKYSFCTDHTVLVQTQGGNSNGALCHFPFLYNNHNYTDCTSEGRRDNMK
      ||     |||     |       ||||| |    |  ||  ||
27  EHKYKAE-EHTVVLTVTG-----EPCHFPFQYHRQLYHKCTHKGRPGPQP

444 WCGTTQNYDADQKFGFCPMAAHEEICTTNEGVMYRIGDQWDKQHDMGHMM
    || || | | ||  | |                        | |   |
71  WCATTPNFDQDQRWGYCLE------------------PKKVKDHCSKHSP

494 RCTCVGNGRGEWTCIAYSQLRDQCIVDDITYNVNDTFHKRHEEGHMLNCT
       |   |    ||                       |
103 ---CQKGG----TCVNM----------------PSGPHCL----------

544 CFGQGRGRWKCDPVDQCQDSETGTFYQIGDSWEKYVHGVRYQCYCYGRGI
    |     |   |     |       |      |          | | |
120 CPQHLTGNH-CQKEK-CFEPQLLRFFHKNEIWYRTEQAAVARCQCKGPDA

594 GEWHCQPLQTYPSSSGP
       ||| |        |
168 ---HCQRLASQACRTNP

        Score : 475
        SW  score : 133
        pattern alignment scores : 34 and 137
```

Fig. 8. Alignment of sequences FA12_HUMAN and FINC_HUMAN with two different weights.

behavior of the number of occurrences of a pattern in a random text.

In this study, we have focused on patterns from the PROSITE databank, but other biological information (e.g. the secondary structure, chemical characterization, …) could be incorporated instead of, or in addition to, the patterns.

Acknowledgements

The authors are much indebted to J.-J. Codani and E. Glémet for the use of LASSAP (Glémet and Codani, 1997) to implement this algorithm in a large scale environment.

References

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

Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J., 1997. Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Res. 25 (17), 3389–3402.

Bairoch, A., Apweiler, R., 1997. The swiss-prot protein sequence data bank and its supplement trembl. Nucleic Acids Res. 25 (1), 31–36.

Barton, G., Sternberg, M., 1987. Evaluation and improvements in the automatic alignment of protein sequences. Protein Eng. 1 (2), 89–94.

Glémet, E., Codani, J.-J., 1997. LASSAP: a large scale sequence comparison package. Comp. Appl. BioSci. 13 (2), 137–143.

Gribskov, M., Homyak, M., Edenfield, J., Eisenberg, D., 1988. Profile scanning for three-dimensional structural patterns in protein sequences. Comput. Appl. Biosci. 4 (1), 61–66.

Hofmann, K., Bucher, P., Falquet, L., Bairoch, A., 1999. The prosite database, its status in 1999. Nucleic Acids Res. 27, 215–219.

Holloway, J.L., Cull, P., 1994. Aligning genomes with inversions and swaps, In: ISMB, pp. 195–202.

Huang, X., 1994. A context dependent method for comparing sequences. In: 5th Annual Symposium CPM, Springer-Verlag, vol. 807 of LNCS, pp. 54–63.

Karlin, S., Altschul, S.F., 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. Proc. Natl. Acad. Sci. 87, 2264–2268.

Kolakowski, L., Leunissen, J., Smith, J., 1992. Prosearch: fast searching of protein sequences with regular expression patterns related to protein structure and function. Biotechniques 13, 919–921.

Lewin, B., 1997. Genes IV. Oxford University Press, New York.

McEliece, R.J., 1977. The Theory of Information and Coding. Encyclopedia of Mathematics and its Applications, vol. 3. Addison-Wesley, Reading, Massachusetts.

Miller, W., Boguski, M., Raghavachari, B., Zhang, Z., Hardison, R., 1994. Constructing aligned sequence blocks. J. Comput. Biol. 1 (1), 51–64.

Needleman, S.B., Wunsch, C.D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453.

Nicodéme, P., Salvy, B., Flajolet, P., 2002. Motif statistics. Theoretical Computer Science, in press.

Nicodéme, P., Salvy, B., Flajolet, P., 1999. Motif statistics. In: ESA '99, Vol. 1643 of Lecture Notes in Computer Science, Springer-Verlag, Proc. European Symposium on Algorithms, Prague, pp. 194–211.

Parida, L., Floratos, A., Rigoutsos, I., 1998. Musca: an algorithm for constrained alignment of multiple data sequences. In: Proceedings 9th Workshop on Genome Informatics, Tokyo, Japan.

Parida, L., Floratos, A., Rigoutsos, I., 1999. An approximation algorithm for alignment of multiple sequences using motif discovery. J. Comb. Optimization 3 (2/3), 247–275.

Posfai, J., Szaraz, Z., Roberts, R., 1994. Visa: visual sequence analysis for the comparison of multiple amino acid sequences. Comput. Appl. Biosci. 10 (5), 537–544.

Smith, R., Smith, T., 1992. Pattern-induced multi-sequence alignment (pima) algorithm employing secondary structure-dependent gap penalties for comparitive protein modelling. Protein Eng. 5, 35–41.

Smith, T.F., Waterman, M.S., 1981a. Comparison of biosequences. Adv. Appl. Math. 2, 482–489.

Smith, T.F., Waterman, M.S., 1981b. Identification of common molecular subsequences. J. Mol. Biol. 147, 195–197.

Wilbur, W.J., Lipman, D.J., 1984. The context dependent comparison of biological sequences. SIAM J. Appl. Math. 44, 557–567.