SSMAL: similarity searching with alignment graphs*

Pierre Nicodème†

INRIA-Rocquencourt and LIX École Polytechnique

Abstract

Motivation

We want to provide biologists with a fast and sensitive scanning tool for searching local alignments of a protein query sequence against databases of protein multiple alignments such as ProDom. Reversely, we want to provide a tool for locally aligning a protein multiple alignment query against a protein database such as SWISSPROT.

Results

We developed the program SSMAL (Shuffling Similarities with Multiple Alignments) that utilizes features of the Blast (Alschul et al., 1990) algorithm and part of the Blast code. Our software allows both scanning multiple alignments and searching with a multiple alignment. Deletions in the multiple alignment only are handled and a SSMAL search may miss some similarities found by a profile search. However a SSMAL scan of a database as ProDom would be 20 to 30 times faster that a profile scan. In the worst case a SSMAL search is about 9 times faster that a profile search.

Availability.

http://www.dkfz-heidelberg.de/tbi/people/nicodeme and follow the hyperlink SSMAL.

Contact.

Email: p.nicodeme@DKFZ-Heidelberg.de

Introduction

We provide a method for a search of protein sequences against multi-alignment databases such as ProDom (Sonnhammer & Kahn, 1994)

Scanning ProDom for similarity search is usually done by searching similarities against a database of consensus sequences of the multi-alignments.

We developed a new approach, different to profiles (Gribskov, 1994) (Gribskov et al., 1987) and using alignment graphs built on a distinction between well-conserved and weakly-conserved regions. Alignment graphs have previously been used by Hein (Hein, 1989; Hein, 1990) to align homologous sequences, given their phylogeny, and by Schwikowski and Vingron (Schwikowski & Vingron, 1997) to handle the generalized tree alignment problem.

The biological intuition underlying our approach relies upon the hypothesis that the variable subsequences composing the weakly-conserved regions of a multi-alignment have less structural constraints and may mutate around a skeleton built over the well-conserved parts of a multi-alignment. The algorithm therefore rests on the idea that an alignment with a multiple alignment must in the strongly conserved regions match the consensus of this multiple alignment while it may match any of the sequences of the multiple alignment in the weakly conserved regions. There, it will possibly match different sequences (shuffling) in different regions.

System and Methods

SSMAL is written in the C language (Kernighan & Ritchie, 1978). It has been tested on a Sun Solaris platform. Its main features are:

- Preparation of the database - equivalent to BLAST setdb.
Fig. 1. An Alignment-Graph and a Best-Path. In the conserved regions (in solid lines), we take the consensus. We build branches in the weakly-conserved regions (in dashed lines). Combination of these branches is allowed by the algorithm, producing a best path (in bold in the left part of the figure). Note that the example of the right part of the figure corresponds to exact matching while our algorithm performs as Blast approximate matching, by use of a similarity matrix. In this example, B, F and C are supposed to be weakly similar, while E and Z are strongly similar.

- Poisson clumping-declumping probabilistic calibration of the multi-alignments and printing of the parameters (λ and K) for the multi-alignments
- Search for similarity of a query sequence against a database of multi-alignments.
- Search for similarity of a query multiple alignment against a database of sequences.

Alignment-Graphs and Best-Path method

Overview of the method

Like in the Blocks approach (Henikoff & Henikoff, 1991), we consider that the well-conserved and aligned regions are joined by weakly-conserved regions, although in SS-MAL we consider this in a different way. We take the consensus of the sequences as representant for the well-conserved regions while using a set of “parallel” subsequences to represent the weakly-conserved regions. When the alignment algorithm processes positions in a weakly-conserved region, the subsequence with the best match is selected.

We describe in the following section how we discriminate between conserved or non conserved positions to build conserved regions; we also discuss heuristic choices made to keep the method efficient and to prune alignments without real significance ("noisy hits").

The Alignment-Graph of a multi-alignment then is composed of subsequences of the consensus and of sets of parallel subsequences (branches) in the weakly-conserved regions (branch regions as opposed to consensus regions). See Fig. 1 for a schematic example. In the context of an alignment process, an admissible path never intersects two branches of the same branch region.

Fig. 1 shows on the right side a sketched multi-alignment, and on the left side the corresponding schematic Alignment-Graph.

Considering now a similarity matrix \( M \), the score of an alignment of an admissible path with amino-acids \( p_1, p_2, \ldots, p_k \) and of a subsequence with amino-acids \( q_1, q_2, \ldots, q_k \) of a query sequence is the sum of the scores:

\[
S = \sum_{i=1}^{k} M[p_i, q_i].
\]

We call the admissible path with optimal alignment score \( S \) with respect to the query the Best-Path.
Getting the well-conserved and weakly-conserved regions from the multi-alignment

Definition of a coefficient of conservation for a multi-alignment position

We now describe how we discriminate between well-conserved and weakly-conserved positions.

Definitions: Given a multi-alignment of n sequences of length \( l \), let \( a_{ij} \) be the amino-acid at position \( i \) in the sequence \( j \). Let \( s(ab) \) be the score of similarity of two amino-acids \( a \) and \( b \) in the similarity matrix \( \mathcal{M} \). Let \( c_i \) be the consensus amino-acid at position \( i \), such that \( \sum_{k=1}^{n} s(a_{ik}c_i) \) is maximal. Let \( \overline{c_i} \) be the anti-consensus amino-acid at position \( i \), such that \( \sum_{k=1}^{n} s(a_{ik}\overline{c_i}) \) is minimal. To normalize as much as possible the level of similarity at a position, we define a coefficient of conservation \( \tau_i \) at position \( i \) as follows:

\[
\tau_i = \frac{1}{n} \sum_{k=1}^{n} \frac{s(a_{ik}c_i) - s(\overline{c_i}\overline{c_i})}{s(c_i c_i) - s(\overline{c_i}\overline{c_i})}. \tag{1}
\]

By construction, \( \tau_i \) is positive in the range \([0, 1]\); for a perfectly conserved position, we have \( \tau_i = 1 \), and \( \tau_i \) decreases as the level of similarity does.

We then choose a threshold value \( \hat{\tau} \), which is constant for the multi-alignment. Depending on whether or not \( \tau > \hat{\tau} \), we say that the position is conserved or not. Different strategies may be adopted when considering a set of multi-alignments: choosing an unique value for all the multi-alignments, or choosing a \( \hat{\tau} \) in relationship with the number of sequences in the multi-alignment.

Statistical results show that the empirical choice of \( \hat{\tau} = 95\% \) for all ProDom multi-alignments, with BLOSUM62 as similarity matrix \( \mathcal{M} \), is reasonable.

Mathematical model

Simulations show that the Karlin-Altschul model (Karlin & Altschul, 1990) applies for SSMAL. With \( l = l_q \times l_t \), where \( l_q \) is the length of a query sequence, \( l_t \) is the length of a target multi-alignment and \( S(l) \) the score of the optimal alignment (see end of section ), we have

\[
\Pr \left( S(l) > \frac{\log(l)}{\lambda} + x \right) = 1 - e^{-K \lambda x}, \tag{2}
\]

for some constants \( K \) and \( \lambda \).

However, a technical condition of validity of this model is that the average score at a position is strictly negative. The average score at a position increases as the number of branches increases, drifting to positives values. Analytical and numerical analysis done on an approximate model of normal distributions for the scores of random branches against random sequences show that limiting the number of branches to 5 and eliminating very short branches (under 3 positions) suffices to insure a negative average score (see (Nicodeme, 1997)).

This heuristic implies the necessity of collapsing together the most similar branches during the preprocessing step of the algorithm, described in the next section.

A branch region then will be composed of at most 5 branches of at least 3 contiguous positions with \( \tau > 95\% \).

Implementation

Preprocessing step: construction of Alignment-Graphs

This step defines the strongly conserved positions and reduces when necessary the number of parallel branches.

We reduce the set of branches by aggregating together branches with high similarity, and by taking the consensus for each of these sets. We detail how we reduce the number of sets from \( k \) to \( k - 1 \).

We consider a partition of the set of \( n \) branches \( b_i \) in \( k \) sets \( S_j, j = 1 \ldots k \); we have:

- **reduction step:**
  1. take the consensus \( C_j \) of each set \( S_j \);
  2. for each pair \( l, m \), with \( 1 \leq l < m \leq k \), compute the score \( s(C_l C_m) \), as the sum of the score of the individual positions; select the pair \( l', m' \) with highest score;
  3. merge the sets \( S_l \) and \( S_m \).

Initialization of the preceding algorithm is made by considering a partition with a single branch in each set \( S_j \); we iterate the reduction step until \( k \leq 5 \), the heuristic value given in the preceding section.

After this step, branches may contain gaps, but there will never be simultaneously gaps on all the branches for a given position (assuming that this does not occur in the initial multiple alignment).

See Fig. 2 for an example of preprocessing.

Main step: searching for an alignment between a query sequence and an Alignment-Graph

This step differs from Blast by the extension algorithm. First, when a multiple-alignment is considered, a layout of this multi-alignment is made in as many strings...
Fig. 2. A region (positions 270 to 319) of the multi-alignment 1284 of ProDom28, and the corresponding preprocessing output. Positions considered as conserved have a conservation coefficient \( \tau \) such that \( \tau > \hat{\tau} = 95\% \).

as there are branches plus one string for the consensus. Then, inside the branch strings, the positions corresponding to a consensus region are set to Null. Reversely, inside the consensus string, the positions corresponding to branch regions are set to Null (see Fig. 4).

Hits detected by the Aho-Corasick (Crochemore & Rytter, 1994) multi-string automaton on high scoring small words are right and left extended for high scoring alignments of the multi-alignment and of the protein sequence (Fig. 3). This extension works similarly to the Blastp extension, with the addition of interruptions when a null byte is hit.

We detail in Fig. 4 the left extension; right extension is symmetrical.

Applications

Test-1: SSMAL search. Comparisons of different methods

Given a family of homologous proteins sharing sufficient similarity to build a multi-alignment, we want to detect in a sequence database which ones are related to the family.

Fig. 6 presents a synthetic comparison of results of similarity search with 6 different methods. This test does not consider gaps.

---

**HIT-EXTENSION()**

begin

BLASTP-LEFT-AND-RIGHT-EXTENSION()

until hit Null-sentinel-byte

(initializes rightscore and rightsum)

forever do

EXTEND-LEFT()

if RecordScore > rightscore

and -rightscore > -DropOffScore

and rightsum > -DropOffScore then

leftscore = RecordScore

leftsum = ExtendScore - RecordScore

EXTEND-RIGHT()

if RecordScore <= leftscore

or -leftscore <= -DropOffScore

or leftsum <= -DropOffScore then break

else

rightscore = RecordScore

rightsum = ExtendScore - RecordScore

fi

else break fi

od

if RecordScore high

then MEMORIZE-ALIGNMENT() fi

end

---

**Fig. 3. SSMAL: extension of a Hit.**

- (A) SSMAL (Best-Path method with Alignment-Graph).
- (B) Unweighted Consensus.
- (C) Weighted Consensus.
- (D) Unweighted Profile (profile with average score at each position).
- (E) Weighted Profile (profile with weighted score at each position).
- (F) Set of sequences: with \( n \) sequences \( S_m \) in the multi-alignment, for each test sequence \( S_i \), the sequence \( S_m \) scoring best with \( S_i \) is selected, and the corresponding maximal score memorized.

**Weighting schemes**

Weighting schemes for consensus and profile of sequence alignments have been computed by using ClustalV (Higgins et al., 1992) and TreeWgt (Gerstein et al., 1994).
Example of lay out of a multi-alignment for extension (C: conserved U: unconserved)

type = "CCUUUCCCCCUCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC"
consensus = "0KV000GAGGGV000000000QQ0000000QHRSVTPII000D0000000000"

branch[0] = "0004V000000QGALAFLALKOMQIPIASE00000000000LYV0INEAMDKAPO"
branch[1] = "0001TIV000000GS3YAFILNONQGIADE000000000000LIAVOIMKAMDKTGO"
branch[2] = "000LYV0000004YSLYRIAN0GNNPGK000000000000LHLOQAMQVDGO"
branch[3] = "0004V0000000SMHLPFLAS0GEVGQ00000000000KLLOSERSFQALEG0"
branch[4] = "000AILO000000GQPLSLLKHEHVE0000000000LALYOIEMDAANGO"

EXTEND-LEFT()
begin
  forever do
    if ExtensionScore - RecordScore < -DropOffScore then return fi

    OldRecordScore = RecordScore
    OldExtensionScore = ExtensionScore
    ExtensionScore = -Infinity

    if Type-of-Region = "branch (U)" then
      foreach branch of the region do BLAST-EXTEND-LEFT() od
    else
      for consensus of the region do BLAST-EXTEND-LEFT() od
    fi

    if Beginning-of-region not reached then break fi
    if Beginning-of-multialignment reached then break fi
    if Beginning-of-protsequence reached then break fi
  od
end

BLAST-EXTEND-LEFT()
begin
  score = OldRecordScore
  sum = OldExtensionScore - OldRecordScore

  forever do
    sum += ScoreMatrix[protsequence][multialignment]
    if sum > 0 then
      Memorize-Position-as-Putative-Beginning-of-Alignment
      score += sum
      sum = 0
    fi
    if Hit-Null-Sentinel-Byte then break

    x = -score
    if x < -DropOffScore then x = -DropOffScore fi
    if sum < x then break fi
  od
  if score > RecordScore then RecordScore = score fi
  if score + sum > ExtensionScore then ExtensionScore = score + sum fi
end

Fig. 4. SSMAL: lay-out with Null bytes and left extension.
Fig. 5. \(Z'\)-scores for SSMAL and Weighted Profiles

Fig. 6. The best 30 \(Z'\)-scores obtained with a globin Probe-Family from a target set of 66 globins and 1875 non globins; (A) SSMAL; (B) Unweighted Consensus; (C) Weighted consensus; (D) Unweighted Profile; (E) Weighted Profile; (F) Set of Sequences. Non Globins are represented by a black rectangle. Letters -A,...,F indicate the method giving the highest \(Z'\)-score for the corresponding protein.
ClustalV produces a binary phylogenetic-type tree for sequences belonging to a multi-alignment; each sequence is assigned to a node in the tree and the tree topology gives the similarity level between sequences. With such a topological tree as input, TreeWgt gives weights for the sequences, and these weights may be used to compute weighted consensus or weighted profiles.

Description of the tests

Each sequence with odd rank in multi-alignment #1 (globin family) of ProDom28 is selected to build a multi-alignment, the Probe-Family.

Test sequences are:

- globin sequences of SwissProt32 which do not belong to multi-alignment #1 of ProDom28 (66 sequences);
- non globin sequences randomly chosen inside SwissProt32 (1875 sequences).

For each method, $Z'$-scores are computed as follows.

1. For each test sequence, the alignment with the Probe-Family producing the best score $S$ is considered.
2. Expectation $E$ and variance $V$ of these scores for the non-globin sequences are computed.
3. A $Z$-score is computed as
   \[ Z = \frac{1}{V(S)}(S - E(S)). \]
4. Equation 2 shows that linear regression applied to the plot $Z = f(\log(L))$, where $L$ is the length of the sequences, gives a line with slope $\frac{1}{\lambda}$. Computing $\lambda$ by this linear regression, we define a $Z'$-score as $Z' = Z - \frac{1}{\lambda}\log(L)$, to get rid of the deviation caused by the sequence lengths.

Although the $Z'$-scores most likely follow an extreme value distribution, an approximation by a normal distribution was used simply for the convenience of comparing the scores, not as statistical indicator.

Discussion of the plots

Fig. 6 gives a synthetic comparison of the test results for the six methods. The proteins with the 30 best $Z'$-scores are represented in order of decreasing $Z'$-scores, a shadow white rectangle representing a globin and a black rectangle representing a non-globin (false positive). Moreover, for each protein in this figure, the method producing the highest $Z'$-score is given in the shadow rectangle. Weighted Profiles are the most sensitive method, but SSMLAL detects several similarities (marked "-A") which are not detected by the other methods. SSMAL is also the method with the smallest number of false positives in these 30 best scores.

When looking more precisely at the results, SSMAL detects similarities of the globin Probe-Family with globins GLB2_TYLHE, GLBT_CHITH, GLB1_CALSO, GLB1_LUMTE and GLB1_PHESE which are not detected by the other methods (GLBT_CHITH is an insect globin whose similarity with the Probe-Family is distant). Similarities with globins GLB2_CALSO and LGB3_SESRO are better detected by SSMAL. The specific differences between SSMAL and the weighted profile results can be explained from the composition of the Probe-Family. The Probe-Family consists of 304 globins, 27 of which being globins subunits (GLB...), 241 being $\alpha$ and $\beta$-chains (HBA...,HBB...), and 36 being myoglobin (MYG...). Although the GLB globins have higher weights, the weighted profile favors similarities to the dominant $\alpha$ and $\beta$-chains. Some leghemoglobins have good similarities with both the $\alpha$ and $\beta$-chains; these similarities are not detected either with the $\alpha$-chain nor with the $\beta$-chain separately. The preprocessing step of SSMAL, on the other hand, builds set of branches respecting the diversity of the Probe-Family; this gives a chance for similarities to the subunits globins GLB which are well detected by SSMAL and not by the weighted profiles. See (Nicodème, 1997) for further results.

Test-2: SSMLAL search. Comparison with generalized profiles

The tests described below have been performed using a Sun Ultra-Sparc.

This test compares a SSMLAL search and a generalized profile search (Bucher et al., 1996). Gaps are allowed. The Probe-Family consists of 30 sequences from the lactate test family given by Bucher. The test sequences are:

- 48 lactate sequences, 47 of which belonging to the lactate test family of Bucher;
- 1872 random sequences from SWISSPROT32 which are not lactates.
Krogh and al. (Krogh et al., 1994) have experimentally shown that generalized profiles and profiles-HMMs are equivalent. Bucher and al (Bucher et al., 1996) gave theoretical demonstration of this equivalence. Comparisons made in this paragraph with generalized profiles therefore stand also as comparisons with profiles-HMMs.

SSMAL results

SSMAL search finds 43 out of the 48 target lactates and misses 5. One of these 5 sequences is a very short sequence (MDHC_HUMAN, 14 positions). 3 other false negatives are significantly shorter (34 to 74 positions) than the average size of the target sequences (330 positions). There are false positives at rank 37, 39, 44 and 46. The search time is 8.53 sec..

Generalized profiles

The generalized profile built over the Probe-Family misses the MDHC_HUMAN sequence too, but finds the 47 other ones. False positives are at rank 41 to 45 (5), 49, 50, 53 and 55. The search time is 75.64 sec..

Blastp search with the Probe-Family consensus

This search misses 9 sequences, 8 of which are short sequences (under 40 positions). The search time is 1.35 sec..

SSMAL scan of multi-alignments

We now describe the second application of our method, which consists in looking for similarities of a protein query sequence against a collection of multi-alignments. This approach has a direct application to the database ProDom.

Probabilistic calibration of the multi-alignments

To allow pertinent comparisons of the results across different multi-alignments, the parameters $K$ and $\lambda$ in equation 2 must be computed for each multi-alignment, which corresponds to the calibration step for blocks. It is not possible to do this analytically, but the clumping-declumping method of Waterman and Vingron (Waterman & Vingron, 1994) applies and gives good approximations for $K$ and $\lambda$. No position only consists of gap characters, and a clump therefore is a “diagonal” segment in the comparison matrix of the query and the consensus. Thus declumping is easy. We count the number of clumps $C(t)$ over a score $t$ to check the fit with the Poisson model (Fig. 7) and to compute parameters $K$ and $\lambda$.

The whole calibration of ProDom33 takes 4 hours on a Sun Ultra-Sparc.

Fig. 7. SSMAL. Fit with the Poisson model (5000 simulations with alignment 437 of ProDom28). $C(t)$ is the number of clumps of score over $t$ ($t = 40$). The circles correspond to a Poisson of parameter 1.48

Fig. 8. Result of a SSMAL scan query against ProDom33; the query sequence is the ATP-binding protein MDL_ECOLI.
For searching similarities of a multi-alignment against a collection of sequences, our approach is not as sensitive as profiles, but much faster. For scanning a collection of multi-alignments, we are not as fast as a scan against the consensus, but much more sensitive.

Acknowledgment. I am grateful to Daniel Kahn, Florence Carpin and Jean-Marc Steyaert for frequent discussions about the algorithm proposed in this paper. I received also very efficient help of the Theoretical Bioinformatics group at DKFZ to improve SSMAL.

References


