Automated protein sequence database classification. I. Integration of compositional similarity search, local similarity search, and multiple sequence alignment

Jérôme Gracy¹ and Patrick Argos²

European Molecular Biology Laboratory, Postfach 102209, Meyerhofstrasse 1, 69012 Heidelberg, Germany

Received on November 19, 1996; revised on October 23, 1997; accepted on October 29, 1997

Abstract

Motivation: Genome sequencing projects require the periodic application of analysis tools that can classify and multiply align related protein sequence domains. Full automation of this task requires an efficient integration of similarity and alignment techniques.

Results: We have developed a fully automated process that classifies entire protein sequence databases, resulting in alignment of the homologous sequences. The successive steps of the procedure are based on compositional and local sequence similarity searches followed by multiple sequence alignments. Global similarities are detected from the pairwise comparison of amino acid and dipeptide compositions of each protein. After the elimination of all but one sequence from each detected cluster of closely related proteins, the remaining sequences are compiled in a suffix tree which is self-compared to detect local sequence similarities. Sets of proteins which share similar sequence segments are then weighted according to their closeness and multiply aligned using a fast hierarchical dynamic programming algorithm. Computational strategies were devised to minimize computer processing time and memory space requirements. The accuracy of the sequence classifications has been evaluated for 12,462 primary structures distributed over 341 known families. The percentage of sequences with missed or incorrect family assignments was 6.8% on the test set. This low error level is only twice that of the manually constructed PROSITE database (3.4%) and is substantially better than that found for the automatically built PRODOM database (34.9%).

Availability: The resulting database, called DOMO, is available through database search routine SRS at Infobiogen (http://www.infobiogen.fr/srs5/), EBI (http://srs.ebi.ac.uk:5000/) and EMBL (http://www.embl-heidelberg.de/srs5/) World Wide Web sites.

Contact: gracy@infobiogen.fr

Introduction

The huge accumulation of raw genetic sequences in large databases has provided the need for fully automated computing tools to analyze them. One basic prerequisite for more elaborate data mining involves the accurate classification of available protein sequences into homologous families through systematic searches for similarities and subsequent alignment of the related primary structures.

Recent attempts at database classification have followed two principal directions. The first is based on a manual determination of motifs delineating well-conserved residues within aligned sequence segments of a given protein family. The most well-known database constructed from this perspective is PROSITE (Bairoch et al., 1996). Classification based on such sequence fingerprints requires a manual design of each motif which should be updated in response to newly determined family sequences. Consequently, considerable efforts will be required to accommodate large databases. Recent steps have been undertaken toward the automation of motif determination and calibration, leading to databases such as BLOCKS (Henikoff and Henikoff, 1994b) and PRINTS (Attwood and Beck, 1994). However, their completeness is limited (1167 families in PROSITE 13.0, 932 in BLOCKS 9.3, 600 in PRINTS 13.0) and no comprehensive sequence alignments (if at all) are given.

The second direction tackles the classification task more systematically with an exhaustive search for local similarities followed by an alignment of the detected protein family sequences. The collections DARWIN (Gonnet et al., 1992), HHS (States et al., 1993), PRODOM (Sonnhammer and Kahn, 1994) and that of Smith and Smith (1990) have been constructed from this perspective. However, these ap-
approaches for entire databases are prohibitively slow, sometimes involving several weeks of computer processing (Gonnet et al., 1992), and the resulting classifications are often too narrow with families limited to closely related sequences [see Table 1 in the accompanying paper (Gracy and Argos, 1998)]. To overcome these limitations, it is suggested here that the analysis process incorporate a cohesive integration of three principal techniques in sequence comparison: compositional similarity search, local sequence similarity search and multiple sequence alignment.

Table 1. Percentages of true positives, false positives and false negatives averaged over all families in a reference set according to PROSITE, PRODOM and the method described in this work

<table>
<thead>
<tr>
<th>Method</th>
<th>True positives</th>
<th>False positives</th>
<th>False negatives</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROSITE</td>
<td>96.6%</td>
<td>1.8%</td>
<td>1.6%</td>
</tr>
<tr>
<td>This work</td>
<td>93.2%</td>
<td>0.3%</td>
<td>6.5%</td>
</tr>
<tr>
<td>PRODOM</td>
<td>65.1%</td>
<td>0.9%</td>
<td>34.0%</td>
</tr>
</tbody>
</table>

Compositional similarity search aims to detect related sequences directly from the comparison of the overall distributions of either their amino acid types (Nishikawa et al., 1983), their oligopeptides (Jones et al., 1992; Petrilli, 1993) or their physicochemical properties (Hobohm and Sander, 1995). At the cost of some sensitivity, the comparisons over all database entry pairs can be greatly accelerated since similarity is directly measured by correlation of compositional vectors pre-calculated for each sequence.

Increased sensitivity can be obtained with local similarity searches. The most widely used methods are based on hashing techniques (Altschul et al., 1990). Promising approaches with more refined search strategies have recently been proposed utilizing suffix trees (Lefevre and Ikeda, 1994) as generalizations of hashing tables such that subsequences that share a prefix are grouped into unique subtrees.

To compare full sequences with gaps, the basic dynamic programming algorithm has been adapted in various ways that allow two (Needleman and Wunsch, 1970), three (Mura- ta, 1990) or more (Thompson et al., 1994) sequences to be aligned. Among the numerous improvements that have been developed, three are particularly relevant to this work and involve searches for optimal local alignments (Smith and Waterman, 1981), and reduce the calculation time of alignments with linear gap penalties (Gotoh, 1982) and the space consumption in effecting pairwise alignments (Myers and Miller, 1988).

Here we present an analytical procedure that integrates in successive steps the three main processes of sequence comparison. The most closely related sequences are first detected by pairwise comparison of protein sequence residue and dipeptide compositions. After the elimination of all but one protein in each cluster of very similar members, the selected sequences are compiled into a suffix tree where similar paths point to subtrees sharing possible local similarities which are subsequently checked. Eventually, a multiple alignment is constructed for each cluster of related sequences.

**Algorithm**

**Overview**

The analytical procedure used here involves the following steps (Figure 1).

1. Composition similarity search. Two proteins are considered related if their amino acid and dipeptide composition distances are below pre-defined thresholds depending on their sequence lengths. The summation of compositional distances over residue types is terminated when either the residue or the dipeptide composition threshold is reached. Computational speed has been optimized by appropriate ordering of the terms in the distance computation according to amino acid type. Only one protein is selected in each cluster of very similar members where aligned sequence pairs share at least 65% identically matched residues.

2. Local similarity search. The sequences remaining after step (1) are compiled into a suffix tree which is compressed into a 'position-end-set' graph and self-compared with a depth-first search to detect subtree pairs grouping subsequences sharing comparable prefixes. Their collected leaves provide the locations and proteins

**Fig. 1.** Outline of the analysis process described in this and the accompanying article.
with similar oligopeptides which are extended without insertion and checked for further relationship with a composite criterion combining statistical significance, similarity agglomeration and restricted dynamic programming, all with coefficients optimized on a representative set of proteins with known families. The tree search is accelerated by appropriate ‘branch-and-bound’ tests.

(3) Domain delineation. The detected local and similarities are clustered into anchors composed of multiple, gapless and similar protein sequence segments. Using these anchors, each sequence is then split into domains. This step is not described here and is addressed in the accompanying paper (Gracy and Argos, 1998).

(4) Multiple sequence alignment. The protein domains sharing at least one detected similarity are clustered in the same family and aligned multiply by inserting gaps with a fast hierarchical dynamic programming algorithm. The initial evaluation of pairwise protein similarities by sequence alignment has been accelerated by quick construction of an approximate multiple alignment based exclusively on the detected local similarities. To correct for possible over-representation of strongly homologous protein clusters, each aligned sequence is weighted according to its similarity with others. Final profiles for given sequence families are calculated from the concatenation of partial multiple sequence alignments between consecutive local similarity regions utilized as mandatory anchors.

**Compositional similarity search**

The comparison of amino acid and dipeptide compositions over protein sequence pairs permits fast similarity searches since the number of elementary operations is independent of sequence length. Let \( \{l_A(i), n_B(i)\} \) and \( \{n_A(i), n_B(i)\} \), respectively, be the numbers of amino acids of type \( i \) and of consecutive dipeptides of type \( (i,j) \) in the sequences \( A \) and \( B \). The compositional distances are then calculated by:

\[
d_1(A, B) = \sum_{i=1,20} |n_A(i) - n_B(i)| \quad \text{and} \quad \frac{d_2(A, B)}{d_1(A, B)} = \sum_{i=1,20} \sum_{j=1,20} |n_A(i,j) - N_B(i,j)|.
\]

In comparing systematically all the sequences of a database, the vast majority of sequence pairs will not be related. Since the terms of the distances \( d_1 \) and \( d_2 \) are positive, a dramatic speed-up can be obtained if additive calculations are terminated when the partial sums over residue types exceed a given threshold. The amino acid types \( j \) in each sequence \( A \) can be ordered following the index \( \{k_A(i)\}_i \) from 1 to 20. Following this order, the partial distances \( d_{1,k}(A, B) = \sum_{i=1,20} \frac{|n_A(i) - n_B(i)|}{|l_A(i)|} \) are calculated by successive addition such that the residue types observed in \( A \) with an unexpected low or high frequency contribute initially to the distance calculation and enhance the chance of reaching the threshold for dissimilarity in the sequence pairs. Similarly, a dipeptide index \( \{l_A(i), m_A(i)\}_i \) can be built for each sequence such that the quantity \( |n_A(l_A(i), m_A(i))/l_A(i) - f(l_A(i), m_A(i))| \) is uniformly decreasing and the partial distances are calculated as \( d_{2,k}(A, B) = \sum_{i=1,20} \frac{|n_A(l_A(i), m_A(i)) - n_B(l_A(i), m_A(i))|}{l_A(i)} \). The indexes \( k_A \) and \( l_A \), and the various \( m_A \) and \( n_A \) terms, need only be calculated once for each sequence before starting the evaluation of the pairwise distances.

The determination of appropriate distance thresholds above which a pairwise similarity is rejected will be detailed in the Results. The sequences sharing at least one accepted similarity are grouped and only one sequence in each cluster is selected for further steps. This fast selection reduces the complexity of the subsequent local similarity search step.

**Local similarity search**

**Suffix tree construction.** The detection of local similarities is based on the construction of a suffix tree (Aho et al., 1974), which is rooted and groups into each of its subtrees all the subsequences of a protein set that share a common N-terminal substring (prefix) given by the sequence of node labels along the path from the root of the tree to that of the subtree (Figure 2). The paths from the root of the tree to its leaves correspond to protein subsequences which do not contain a repeated prefix shared by a given protein set. The protein identification and the last position of one such subsequence are labeled on a corresponding leaf. A suffix tree does not contain rooted paths with the same sequence of labels.

---

**Fig. 2.** Suffix tree corresponding to three proteins (P1, P2, P3) with the respective oligopeptide sequences CDC, CDCD and CAD. A sequence of connected labels (amino acid single-letter codes) from the root to an internal node corresponds to an actual repeated amino acid subsequence, while the protein identification and last numbered position of the subsequence are indicated by labeled leaves. The tree contains duplications of leaves and subtrees highlighted by differently shaded leaves and nodes.
The position-end-set graph, a compressed suffix tree where all the suffixes that share the same ending residue position are clustered, is built according to an algorithm developed by Lefevre and Ikeda (1993) where the amino acids of the protein set are processed sequentially, thereby resulting in an iterative update of the end-set equivalence classes of the tree. To reduce the size of the graph further, we have used a simplified residue alphabet of 12 letters obtained by clustering the amino acids with closest Gonnet substitution scores (Gonnet, 1992): a = {A,S,T}, c = {C}, d = {D,E}, f = {F,Y}, g = {G}, h = {H}, i = {I,L,M,V}, k = {K,R}, n = {N}, p = {P}, q = {Q}, w = {W}, where the capitals correspond to the 20 amino acid single-letter codes. The low-complexity segments of the considered protein sequences, whose relative similarities could be over-estimated due to their strong compositional bias, are initially detected and masked using the procedure described by Claverie and States (1993). This prevents assignments of false similarities between segments sharing the same densely repeated amino acids.

Searching for local similarities. We have adapted the algorithm described by Lefevre and Ikeda (1994) to extend further the usage of a suffix tree in fast searches for local and ungapped pairwise similarities including amino acid mismatches. The detection of nodes corresponding to similar protein fragments is based on a depth-first self-comparison of the tree: each time a node is reached, it is transversally compared with every node of the same depth and a similarity score is calculated; the node’s direct successors are then processed. The similarity score of two nodes with the same depth is the accumulated substitution score of the matched residue pairs successively found along the two paths that bind the nodes to the tree root. When one of the two compared nodes is a leaf, the leaves of the subtree rooted by the other node are collected and the two corresponding subsequence sets are checked for relative similarities as subsequently explained.

This matching algorithm can be described by the following recursive routine written in pseudo code:

```
MatchTrees(node1, node2, score, l, l1, l2)
if score < ScoreLowerBound(l) then return
if node1 is a leaf then
    if node2 is a leaf then CheckSimilarity(protein(node1), protein(node2), score, l, position(node1) - l1, position(node2) - l2)
    else for each successor suc2 of node2 do MatchTrees(node1, suc2, score, l, 1, l2 + 1)
else if node2 is a leaf then for each successor suc1 of node1 do MatchTrees(suc1, node2, score, 1, l1 + 1, l2)
else for each successor suc1 of node1 do MatchTrees(suc1, suc2, score + m(aa(suc1), aa(suc2)), 1 + 1, 1, 1)
```

The similarity search is initiated by calling MatchTrees(root, root, 0, 0, 0, 0). The quantity \( m(aa(suc_1), aa(suc_2)) \) corresponds to the substitution score of the amino acids given by the labels of nodes suc_1 and suc_2. The substitution matrix used here was based on BLOSUM62 (Henikoff and Henikoff, 1992) following the conclusion of Pearson (1995) that this matrix is the most accurate for local similarity detection. The substitution matrix BLOSUM62-12 for the 12-letter alphabet used here was calculated by averaging the BLOSUM62 scores over the cardinal product of each residue class pair. The algorithm complexity is controlled by the cut-off condition score < ScoreLowerBound(l) which determines whether the two paths of length l with similarity score score are to be considered for extension. These cut-offs were optimized using a set of 10 000 pairs of randomly generated segments with lengths 20–50 residues. The values of ScoreLowerBound(l) achieving the optimal balance between search complexity and sensitivity were found to be 2, 4, 7, 9, 11, 13, 16 and 17 for l varying from one to eight, respectively. No cut-offs were used for lengths longer than eight.

Checking the detected similarities. The significance of the similarity score (score) between any two sequence fragments, each of length l and taken from respective proteins protein_1 and protein_2 with respective sequence start positions p_1 and p_2, is examined by the routine CheckSimilarity(protein_1, protein_2, score, l, p_1, p_2). A simple but fast search for the optimal segment bounds is first effected. The segments are extended iteratively and without gaps by one residue towards their N-termini. If the statistical significance of the extended similarity is improved, then the new segment bounds are declared and stored. The longest extensions allowed are 30 residues in length, sufficient for a rough approximation of similarity that can be further refined. A similar extension procedure is then performed towards the appropriate C-termini. Finally, the optimized segment pair is accepted if its statistical significance exceeds a given cut-off.

The raw accumulated similarity score of two protein segments cannot be used directly as a measure of statistical significance as the score depends on the residue length of the compared segments. The a priori probability \( p(s) \) of observing a score s between two segments of length l is obtained from the recursive convolution \( p(s) = P_2 \prod_{l=1}^{s-1} p_1(l) \), where \( p_1(l) \) is the probability of finding a score l based on the BLOSUM62-12 residue class exchange weights, for two randomly chosen amino acids. Because there are \( (N_1 - l + 1)(N_2 - l + 1) \) ways of extracting a segment pair of length l from two proteins of respective lengths \( N_1 \) and \( N_2 \), the probability of observing at least one segment pair of length l whose score is greater than s is given by \( sc_0(s,N_1,N_2) = 1 - (1 - q(s))(N_1 - l + 1)(N_2 - l + 1) \) which to the first order is approximated by \( sc_0(s,N_1,N_2) = (N_1 - l + 1)(N_2 - l + 1)q(s) \)
if \( N_1 N_2 q(s) \ll 1 \), a condition which holds for significant similarities.

The validity of each local similarity produced by the tree self-comparison under the significance criteria just discussed is further checked in two ways. The first evaluation, called \( sc_1 \), is calculated by adding the scores of all possible and allowed similarities in a given pairwise sequence alignment, including the considered similarity, without the use of gaps longer than 40 residues (Figure 3). Similarities and/or similarity clusters are assembled according to a highest-to-lowest score order.

The second evaluation, called \( sc_2 \), is the Gonnet substitution matrix-based alignment score (Gonnet, 1992) of 100-residue-long sequence segments whose centers are the middle positions of the pairwise similarity (the segments can be shorter if one sequence position of the similarity is <50 residues from a sequence terminus). The latter alignment is built using an adaptation of the dynamic programming algorithm allowing locally optimal alignments to be found (Smith and Waterman, 1981). Linear gap penalties \( o + ge \) with opening cost \( o \) at −10 and extending cost \( e \) at −1 were used; \( g \) is the gap length. Application details and thresholds are given in the Implementation section.

**Multiple sequence alignment**

The next procedure involves building a multiple alignment for each detected family of related sequences. A hierarchical clustering of the related sequences (Thompson et al., 1994) first groups and aligns the closest sequences inferred from a rough initial multiple sequence alignment. Then, following the leaf-to-root ordering in a tree, each sequence or group of aligned sequences is successively added and matched to the accumulating multiple alignment through the use of profiles (Gribskov et al., 1987) and dynamic programming.

**Tree construction.** Tree construction for hierarchical clustering usually requires an estimation of the similarity of every familial sequence pair. To reduce the extensive calculations of pairwise alignments over families, a rough multiple alignment is effected without dynamic programming and is entirely constructed from the previously detected local similarity segments whose central residues constitute anchoring points for multiple alignments (Figure 4).

Positions are first assigned to the anchors such that the distance between two consecutive anchors is the length of the longest sequence fragment that joins them. Shorter sequence fragments between the same two anchors are split into two equal length segments which are bound to their corresponding anchors. This simple approach yielded an acceptable basis in dealing with large databases to estimate the relative similarity of sequences despite potential minor misalignments as exemplified in Figure 4 (M of sequence 1, and I, Q and R of sequence 2).

**Profile construction.** To correct statistical bias due to clusters of strongly related sequences, each protein sequence \( s \) of a multiple alignment receives a weight \( w(s) \) proportional to its contribution to the residue variability of the family, namely:

\[
w(s) = 1/(\text{end}(s) - \text{beg}(s) + 1) \prod_{s = \text{beg}(s), \ldots, \text{end}(s)} 1/n(a_i, p_i)
\]

where \( \text{beg}(s) \) and \( \text{end}(s) \) are the first and last sequence positions of \( s \), \( a_i \) is the \( i \)th amino acid of the corresponding protein, \( p_i \) is the position of \( a_i \) within the current multiple alignment, and \( n(a_i, p_i) \) is the number of amino acids of type \( a_i \) aligned at the multiple alignment column corresponding to position \( p_i \). Noting the number of subsequences in the multiple alignment as \( N \), the resulting subsequence weights vary from \( 1/N \) (100% similarity level) to 1 (0% similarity level). Therefore, a roughly similar weight will be assigned both to a cluster of almost identical sequences and to an isolated protein sharing only very weak similarities with others.

Each intermediate multiple sequence alignment is then summarized by a profile which relates the frequency of each residue type at each alignment position (Gribskov et al., 1987). The pseudo-frequency \( f(a, p) \) of the residue of type \( a \) at the alignment position \( p \) is estimated by weighted residue counts over all subsequences that contain \( a \) aligned at \( p \), i.e.

\[
f(a, p) = 1/W \sum_{s \colon \exists i, a_i = a \text{ and } p_i = p} w(s)
\]

**Pairwise alignment construction.** Since the sequences are aligned at positions corresponding to initially detected local similarities, it is possible both to reduce the alignment com-

---

**Fig. 3.** The three possible scenarios when assembling local sequence similarities. The sequence pairs are each symbolized by two horizontal lines and each detected similarity by a shaded parallelogram. The first position of each similar segment is indicated in italics. (1) The two similarities are assembled since \((70 - 29) - (54 - 35) = 22 < 40\). (2) The two similarities are not assembled since \((87 - 2) - (42 - 17) = 70 > 40\). (3) The two similarities are not assembled since they do not preserve the proper N- to C-terminal order.

**Fig. 4.** An exemplary multiple sequence alignment built from local similarities whose central residues are highlighted by bold characters.
plexity and increase the alignment quality. Thus, whole alignments are constructed by concatenating sequence fragments between consecutive anchors and aligning them by classical dynamic programming (Gracy and Sallantin, 1994). Whereas the number of elementary operations required by dynamic programming is proportional to the product of the lengths of the sequences to be aligned, the splitting operation dramatically reduces the complexity order $O(l_1l_2)$ to $O((a + 1)(l_1l_2/(a + 1)^2))$ in the case, for example, of $a$ anchors regularly distributed along two sequences of lengths $l_1$ and $l_2$. Further, since the alignment is a priori constrained to pass through the local similarities, the alignment quality is less dependent on the choice of gap penalties.

Profiles and/or sequences are pairwise aligned using a dynamic programming algorithm implemented with reduced memory space requirements (Myers and Miller, 1988). We have chosen to use the Gonnet matrix (Bennet et al., 1994) which has been shown to be the most effective in aligning sequences correctly over a wide range of similarity levels (Vogt et al., 1995). Matching a residue to a gap (−) yields a null score. The cost of the creation of a new insertion of $n$ residues is linearly estimated as $g_0 + (n − 1) · g_1$. The parameter values $g_0 = −12$ and $g_1 = −1$ were optimized relatively to a set of loosely related protein sequences with known three-dimensional structures from Vogt et al. (1995). Gaps were not penalized at both the N- and C-termini of protein sequences.

To align long sequences in acceptable time, the area of possible alignment paths has been confined in a central diagonal band (Chao et al., 1992) of the pairwise sequence comparison matrix. The width of this band has been empirically fixed to $|l_1 − l_2| + \max(|l_1|,|l_2|)/4 + 10$ up to a maximum value of 150. For example, aligning two sequences of lengths 90 and 120 yields a band width equal to $|120 − 90| + 120/4 + 10 = 70$.

### Implementation

**Assessment of the inferred classification**

Because of the large size of current protein databases, a systematic assessment of all sequence families by visual inspection of relevant alignments cannot be realized. There is also no standard of truth regarding the correct family classification of large sets of sequences. Many databases provide domain family information, but none of them are fully accurate or complete. Even the most thorough PIR (George et al., 1996) and PROSITE (Bairoch et al., 1996) databases contain some mistakes, and their proposed classifications do not cover all sequences and include many subfamilies which should be eliminated from the reference set of families. To reduce the overestimation of false-positive and the underestimation of false-negative family assignments, it is necessary to build a reference set where families are not redundant and include the largest possible number of members. Such a reference set of domain families and their members was obtained by merging the overlapping PIR and PROSITE families in four phases.

1. The family cross-equivalencies between the PIR release 46.0 and PROSITE release 13.0 databases were determined according to the following criterion: the PIR superfamily $A$ was assigned to the PROSITE family $B$ that maximizes the quantity $|A \cap B| − |A \cup B|$, where the symbols $\cap$, $\cup$ and $\ldots$ denote, respectively, the set intersection, the set union and the cardinal of the contained set, and where two sequence parts are considered identical (just counted once) if they belong to the same protein and overlap >50% of the length of the smallest subsequence in the two proteins considered.

2. The associated family pairs were selected if they verify $|A \cap B|^6 \geq |A \cup B|^5$. Using this selection criterion, the overlap between families $A$ and $B$ will be accepted, for instance, if $|A \cap B| = 4$ and $|A \cup B| < 6$ or if $|A \cap B| = 100$ and $|A \cup B| < 252$. This criterion has no impact on our domain classification, but only helps in building a wider reference set and then improving the accuracy of the statistical evaluation.

3. The reference families were defined as the union of each selected family pair.

4. Families with less than four members or fully included in larger ones were removed.

The resulting data set provided the standard of truth to evaluate the performance of our automated approach. It includes 12,462 sequences distributed over 341 families.

The classifications produced by the procedures described here were compared with the reference sets in the following way. First, each calculated family $A$ was associated with the reference family $B$ that maximizes $tp = fp − fn$, where true positives $tp$ are given by $|A \cap B| − 1$, false positives $fp$ by $|\neg B \cap A|$ and false negatives $fn$ by $|B \setminus A|$. The quantity $\frac{100tp}{(tp + fp + fn)}$ was then calculated for each predicted family and the resulting values were averaged over the test set to estimate the overall fraction (percentage) of true positives. The overall proportions of false positives and false negatives were similarly calculated. Assignment of the same weight to each independent family regardless of its size prevents large clusters, such as globins, kinases or immunoglobulins, biasing the assessment. Families with less than four members have also been removed to avoid statistically poor estimations.

**Fast compositional similarity search**

Owing to the high computational cost of dynamic programming in aligning sequences, we utilized only a subset of families randomly selected in the previously described reference set. The resulting test set was useful in optimizing various
thresholds and was composed of 34 families encompassing 677 sequences. For each protein pair \((A,B)\) in this set with respective lengths \(l_A\) and \(l_B\), the following measures were calculated: (i) the percentage of identities \(id(A,B)\) based on the alignment of the sequences; (ii) the two quantities \(d_1(A,B)/\min(l_A,l_B)\) and \(d_2(A,B)/\min(l_A,l_B)\) based on previously discussed compositional distances (see Methods) with a reduced amino acid alphabet of 14 classes: \{A\}, \{C\}, \{D,E\}, \{F,Y\}, \{G\}, \{H\}, \{I,V\}, \{K,R\}, \{L,M\}, \{N\}, \{P\}, \{Q\}, \{S,T\}, \{W\}.

We then determined the exponent and associated threshold which achieve the best discrimination between the sequence pairs that belong to the same family of the reference set and the others. The condition \(d_2(A,B)/\min(l_A,l_B)^{0.68} < 3.7\) was found to be optimally discriminant. The distributions of \(d_1\) and \(d_2\) suggest an efficient bilevel strategy to detect compositional similarities. The embedded distances \(d_1k(A,B)\) are first accumulated iteratively for \(K\) increasing from 1 to 14 unless the threshold \(1.85\min(l_A,l_B)^{0.68}\) is reached before the completion of the distance summation for a given protein pair. The chosen threshold 1.85 eliminates 97.4% of the unrelated sequences while keeping 84.9% of the homologous pairs. For the remaining sequence pairs whose calculated distance \(d_114(A,B)\) is below the previous threshold, the distances \(d_2k(A,B)\) are accumulated iteratively for \(K\) increasing from 1 to 196 unless the threshold \(3.7\min(l_A,l_B)^{0.68}\) is reached. All the sequence pairs whose full distance \(d_2196(A,B)\) remains below the threshold are accepted as related.

**Tuning the local similarity selection criterion**

The relationship between sensitivity and accuracy for different scoring methods has been investigated using the standard-of-truth reference set of sequences consisting of 341 families distributed over 12,462 primary structures. The individually tested measures were the already discussed scores \(sc_0\), \(sc_1\) and \(sc_2\); their linear combination \(sc_3 = \alpha + \sum_i \beta_i sc_i\); and their quadratic combination \(sc_4 = \sum_i 0.5 \beta_i (sc_i – \alpha)^2\). The optimal coefficients in \(sc_3\) were determined using the Levenberg–Marquardt algorithm (Press et al., 1988) by minimizing the quantity summed over all the similarities \(\chi^2 = \sum (sc_3(i) – \delta(i))^2\) where \(\delta(i) = 1\) if the similarity is correct and \(-1\) otherwise. The most discriminant ellipsoid whose equation corresponds to \(sc_3\) was determined in the same way. Each plot of Figure 5 shows for one scoring method the evolution of a variable cut-off value with the resulting fractions of correct and wrong similarities. We finally selected as similarity criterion the score \(sc_3\) whose discriminant equation is \(0.24 sc_0 + 0.82 sc_1 + 0.98 sc_2 > 212\). This criterion permits the detection of 63.3% of the known similarities with a confidence level of 99.99%.

**Discussion**

Table 1 lists the mean percentages over all reference families of correct, false and missed similarities. The resulting classification is compared with those contained in the PROSITE Release 13.0 (Bairoch and Appweiler, 1996) and PRODOM Release 28.0 (Sonnhammer and Kahn, 1994) databanks. Each PROSITE family is characterized by manually adjusted and relatively short motifs that express constraints on conserved residue positions and their relative sequence spacings. The resulting family members listed have sequence segments that match the residual and spacing constraints expressed by the motifs. The PROSITE accuracies were derived from the data given directly in the DR fields which indicate whether the protein is correctly classified or not. The PROSITE members flagged by a question mark, indicating their doubtful membership despite satisfying the motif definition, were counted as correct. It should be noted that with this evaluation scheme the mean percentage of correctly classified sequences in PROSITE is clearly overestimated since the reference set is by definition limited to families whose sequences are largely annotated in PROSITE. The PRODOM database was created by an automated analysis of the sequence similarities detected by the BLAST algorithm (Altschul et al., 1990) with appropriate thresholds, resulting in the decomposition of the proteins into modular domains and multiple alignment of the sequences or sequence fragments assigned to given families. The evaluation of the PRODOM classification accuracy was determined by associating a family with largest overlap with each of those in the reference set followed by the calculation of correctly and wrongly classified protein percentages as previously described.

It is noteworthy that on average, over all families, 93.2% of the sequences are correctly classified by the present analysis process. Compared to the families of the reference set, only 0.3% of all protein assignments were incorrect and only 6.5% were missing. These results can be compared with those of the manually built PROSITE database (1.8% of false positives, 1.6% of false negatives). Since PROSITE constantly updates the pattern description of its motifs, a blind test on a completely new data set would result in a degradation of the present accuracy of PROSITE. Furthermore, our classification is much more reliable than the family collection proposed by the automatically built PRODOM database (0.9% false positives, 34.0% false negatives). These evaluations should be considered as rough indicators of the accuracy of each database since many homologies are ambiguous.

A detailed analysis shows that about two-thirds of the missed homologies are concentrated in a few families, each having a high proportion of false negatives ranging from 6.5% to –25%. These misclassifications fall into three main types. (i) Subtle structural constraints can tolerate very com-
Fig. 5. Relative evolution of the mean percentages over all reference families of all similarities that were correctly and falsely detected for the following scoring methods: \( sc_4 \) (thin solid line); \( sc_3 \) (long dashed line); \( sc_2 \) (solid line); \( sc_1 \) (dotted line); \( sc_0 \) (dashed line).

Complex patterns of amino acid variations (e.g. EF-hand motif). (ii) Conservation is limited to only three or four key residues (e.g. heme-binding site in cytochromes c or iron-binding site in ferredoxins) or a few disulfide bridges (e.g. EGF domain). (iii) In a few families, sequences have been wrongly split into domains (e.g. the hsp20 family). Cases (i) and (ii) require discrimination of very weak similarities from random noise. We are currently investigating profile-based search techniques to improve sensitivity. Case (iii) is analysed further in the accompanying article (Gracy and Argos, 1998).

Less than 91 h of c.p.u. time on one IP21 processor of a Power Challenge Silicon Graphics were necessary to complete the protein classification as well as multiple alignments of the data set of 58 869 sequences totaling 18 918 125 amino acids obtained from a non-redundant merge of the PIR and SWISS-PROT sequences, each of which are complete and contain <1000 residues. The analysis of this data set is detailed in the following paper. The computer processing time required for the different ‘branch-and-bound’ and ‘divide-and-conquer’ techniques described here appears efficient when compared with the 405 CPU days (processor not specified) indicated by Gonnet (1992) in handling only 8 344 353 residues. Parallel processing algorithms were also implemented for the two initial similarity search steps which are the most time consuming. The chosen calibration of the successive processing steps achieves a balanced trade-off between sensitivity, selectivity and speed.

**Conclusion**

We have presented an automated process aimed at a systematic analysis of the sequence similarities shared by proteins in a complete primary structure database. This process integrates three successive steps based on amino acid compositional similarity search, local residue similarity search and multiple sequence alignment, respectively. Attempts have been made to reach an optimal balance between two seemingly opposed requirements: speed and accuracy.

Speed is essential because there are currently \( \sim 100 000 \) complete protein sequences containing nearly 25 million residues and exponential growth looms in the future from whole genome sequencing efforts. An exhaustive search for protein homologies would involve an examination of nearly 5 billion sequence pairs. This pharaonic task could in principle be handled with the brute force of massively parallel computers. We explored another approach by carefully reducing the search space to an acceptable minimum by removing the redundant information provided by very close homologues, embedding the whole data set in a global graph
where segment duplications are represented by one unique path which shortcuts any search of protein segments that share unpromising similarities, and focusing the multiple sequence alignment constructions on enclosed regions delimited by strong local similarities.

A major requirement in this process is obviously accuracy. The range of algorithmic complexity required for detecting homologies does not vary linearly with the similarity level. It has been shown throughout this work that detecting sequences with similarity in the 80–100% range of identically aligned residues can be achieved for each compared pair on average with only six operations. For the 60–80% identity level, an additional 10 operations are necessary. At the other extreme, in the twilight zone of similarities below 25% identity, even the full complexity of dynamic programming can be insufficient to ensure the accuracy of the homology. This latter observation is confirmed by the study of Vogt et al. (1995) who have emphasized a dramatic fall in the accuracy of sequence alignments obtained by dynamic programming for global similarities below the 30% identity level. It is, therefore, important to adapt the processing algorithm to the target homologies.

Any gain in speed of detection can be utilized to refine further the accuracy of the alignment. The hierarchical architecture of the proposed processing steps is precisely aimed at minimizing computational cost without weakening sensitivity. Nonetheless, various techniques have been used to enhance the quality of the results. We have attempted to exploit the complementarity between the locally strong similarities detected by the suffix tree search and the globally diffuse similarities that were emphasized by dynamic programming. A composite selection criterion based on a combination of local and global similarity scores was also designed and yielded a notable improvement in the precision of the evaluation. Multiple sequence alignment tasks were also split into local matching of segments by using the positional constraints expressed by the previously detected similarities. Furthermore, the sequences were weighed before being aligned to correct for possible phylogenetic bias.

The resulting protein classification has a reliability level that is comparable to the most accurate manually managed family collections. Further, the proposed automated classification procedure does not require any human intervention, and its optimized speed and thresholds make it suitable for periodic analysis of complete genomes. In the accompanying paper, we propose as an extension to this work a new algorithm able to delineate the domain boundaries within a set of proteins from the analysis of their detected local sequence similarities. To refine further the quality of the analysis process, we are currently developing similarity search and alignment methods based on the construction of conserved motif segments derived from familial multiple sequence alignment. A database of aligned protein domains resulting from these efforts is available through the search routine SRS (Etzold et al., 1996) via the Word Wide Web at EMBL, EBI and Infobiogen sites with the respective URLs: http://www.embl-heidelberg.de/srs5/, http://srs.ebi.ac.uk:5000/ and http://www.infobiogen.fr/srs5/.

Acknowledgement

This work was supported by a grant from the European Union Biomed and Human Capital and Mobility Program.

References


