

Multiple alignment of protein sequences with repeats and rearrangements

Tu Minh Phuong^{1,*}, Chuong B. Do², Robert C. Edgar³ and Serafim Batzoglou^{2,*}

¹Department of Computer Science, Posts & Telecommunications Institute of Technology, Hanoi, Vietnam,

²Department of Computer Science, Stanford University, Stanford, CA, USA and ³195 Roque Moraes Drive, Mill Valley, CA 94941, USA

Received April 25, 2006; Revised June 28, 2006; Accepted July 5, 2006

ABSTRACT

Multiple sequence alignments are the usual starting point for analyses of protein structure and evolution. For proteins with repeated, shuffled and missing domains, however, traditional multiple sequence alignment algorithms fail to provide an accurate view of homology between related proteins, because they either assume that the input sequences are globally alignable or require locally alignable regions to appear in the same order in all sequences. In this paper, we present ProDA, a novel system for automated detection and alignment of homologous regions in collections of proteins with arbitrary domain architectures. Given an input set of unaligned sequences, ProDA identifies all homologous regions appearing in one or more sequences, and returns a collection of local multiple alignments for these regions. On a subset of the BALiBASE benchmarking suite containing curated alignments of proteins with complicated domain architectures, ProDA performs well in detecting conserved domain boundaries and clustering domain segments, achieving the highest accuracy to date for this task. We conclude that ProDA is a practical tool for automated alignment of protein sequences with repeats and rearrangements in their domain architecture.

INTRODUCTION

Protein multiple sequence alignments have a wide range of applications in bioinformatics, from determination and classification of protein families (1,2) to identification of evolutionarily constrained regions for aiding protein structural or functional prediction (3,4). In principle, given a set of related sequences, a sequence alignment program identifies and organizes regions shared between some or all of the input sequences, and produces multiple alignments: ordered

columns of homologous letters between the sequences interleaved with gaps. In practice, there are diverse kinds of sequence homologies exhibited within classes of related proteins: whereas some protein families share global sequence similarity, other related proteins often have short homologous regions embedded in longer regions of no conservation (5).

For many biologically important protein families, domains may be repeated or shuffled (i.e. the order of domains may vary within a set of related proteins). Significantly, most proteins in prokaryotes and eukaryotes have multiple domains (6), with the number of domain instances ranging from 2 to >100 (7,8). Although many current aligners address some variants of local protein homology detection (Previous Work), few existing methods attempt to fully automate the alignment of proteins with complicated domain organization. Rather, obtaining meaningful alignments of such groups of proteins often requires competent application of many existing sequence comparison tools combined with a great deal of manual intervention.

In this paper, we present ProDA (protein domain aligner), an integrated alignment methodology designed specifically to tackle the problem of aligning proteins with repeated and shuffled domains. At a high level, the algorithm computes local alignments for every pair of sequences, clusters such alignments into blocks of approximately globally alignable subsequences, determines block boundaries, resolves inconsistencies between pairwise alignments and finally multiply aligns these blocks. Algorithmically, ProDA combines several existing ideas in the literature, such as progressive alignment (9), maximum expected accuracy and probabilistic consistency (10) and efficient extraction of local alignments from a Smith–Waterman table (11–13), with novel heuristics for determining domain boundaries and clustering domains (14).

Although accuracy metrics exist for evaluating traditional alignment approaches, these measures do not assess the ability of the aligner to recapitulate the known domain organization of gold-standard alignments. Thus, we designed an ensemble of new accuracy measures that quantify the ability of an aligner to recover the correct domain organization of a

*To whom correspondence should be addressed. Email: serafim@cs.stanford.edu

*Correspondence may also be addressed to Tu Minh Phuong. Tel: 844 8692133; Fax: 8434 511408; Email: phuongtm@fpt.com.vn

set of hand-annotated multi-domain proteins. On reference set 6 of BALiBASE 2.0 (15), which was specially designed to include protein sequences with complex domain structures, ProDA recovers known domain organizations with higher accuracy than previous methods. We conclude that ProDA is a useful method for automated multiple alignment of multi-domain proteins and represents a significant step towards addressing a problem not solved by traditional protein aligners. Source code, executables and benchmarking scripts for program evaluation are publicly available at <http://proda.stanford.edu>.

Previous work

Global alignment methods implicitly assume that input sequences are related over their whole length with the exception of short insertions and deletions (16). When this assumption does not hold, such as in the case of multi-domain proteins with differing domain architectures, global algorithms at best produce a partial view of the homology between their input sequences. For these cases, several local multiple alignment algorithms have been developed, including DIALIGN (8,17), ITERALIGN (18) and SATCHMO (19). The POA multiple alignment algorithm (20) introduced partial order alignments, which represents homologies between the sequences with a directed acyclic graph. However, these methods still require that homologous regions appear in the same order in each sequence.

The problem of dealing with repeated and shuffled regions in protein sequences is not unique to multiple sequence alignment. Motif finders, for example, are one class of methods that can detect repeats and rearrangements in biological sequences (21–23). These methods construct local multiple alignments from blocks of gap-free sequence fragments. However, motif finders can only find short, well-conserved motifs whose instances typically must have the same length.

Dealing with repeated and shuffled domains also plays a role in the construction of protein domain databases. A common approach for constructing profiles of domain families is to use a local pairwise aligner such as BLAST (24,25) to search for homologous sequence fragments, and then realign the extracted fragments using a global multiple alignment algorithm. MACAW (26) was the first tool to perform this process in a semi-automatic manner—users must decide which fragments to cluster in an alignable block and where to set fragment boundaries. A later method, DOMAINER (27) automated these steps and was used to create early versions of the ProDom database of protein domain families (28); however, the DOMAINER algorithm does not allow for gapped alignments, thus greatly limiting its applicability to distantly related sequences. Mocca (29) is a method for finding repeats in a set of protein sequences but requires a repeat instance to be pre-specified by the user.

Repeat-finding algorithms specialize in the detection and alignment of repeats within a single protein sequence (11,30–32). Typically, repeat finders work by finding suboptimal local alignments of a sequence to itself. Repeat libraries are then constructed from collections of suboptimal alignments. Methods for this latter step include graph clustering (31), heuristic repeat boundary detection (30) and statistical significance analysis of repeat alignments (11,32).

Repeat-finding algorithms operate on single sequences, and thus do not directly handle the problem of repeat detection for multiple sequence data; we return to a discussion of their applicability to multiple alignment later (Results).

For sequences with repeats but no rearrangements, the recent RAlign program applies a novel alignment algorithm that accounts for repeats (33). However, RAlign returns a global alignment of the input sequences, and thus makes no attempt to identify relationships between homologous segments within the same sequence.

To date, the only existing tool that directly addresses the local multiple alignment problem is the ABA program of Raphael *et al.* (12). Given a set of sequences and a set of pairwise local alignments, ABA constructs a weighted, cyclic, directed multi-graph called an A-Bruijn graph to represent the multiple alignment of a set of input sequences. Here, construction of a multiple alignment reduces to construction of the corresponding A-Bruijn graph, where high-multiplicity edges represent aligned protein domains. Though A-Bruijn graphs are flexible enough to represent a large variety of domain organizations, the performance of the actual ABA alignment tool in identifying homology in biological sequences is unknown. In Results, we compare the ABA tool to our own method, ProDA, on real data.

MATERIALS AND METHODS

Algorithm overview

Given m protein sequences $S = \{S_1, \dots, S_m\}$ and a minimum alignment length L_{\min} , ProDA returns a set of aligned regions with length at least L_{\min} . The algorithm consists of seven steps (Figure 1):

Step 1: Generation of pairwise local alignments (PLAs)

Using the pair-HMM shown in Figure 2, compute all local alignments of length at least L_{\min} between each pair of sequences in S . To do this, repeat the following until no local alignment of length at least L_{\min} is found:

- (1) Compute the best local alignment using either a variant of posterior decoding (Algorithm Details) or the Viterbi algorithm. Stop if the best local alignment is shorter than L_{\min} . Otherwise, store the alignment found.
- (2) Mark cells in the dynamic programming matrix corresponding to the best alignment found in substep 1, disallowing them from contributing to future alignments.

Step 2: Inference of repeats from pairwise alignments

Some local alignments found in Step 1 may simultaneously overlap in both sequences, indicating the presence of repeats. Break overlapping alignments into shorter non-overlapping PLAs that putatively correspond to individual repeats. This step can be seen as a post-processing step for Step 1 and does not guarantee finding all real repeats.

Step 3: Generation of a block of alignable sequence fragments

Two sequence fragments are alignable if they align to each other in one PLA or both of them align to a third fragment in two different PLAs. A block of alignable fragments is a set of sequence fragments in which at least one fragment is alignable to all the other fragments. Compute the block B

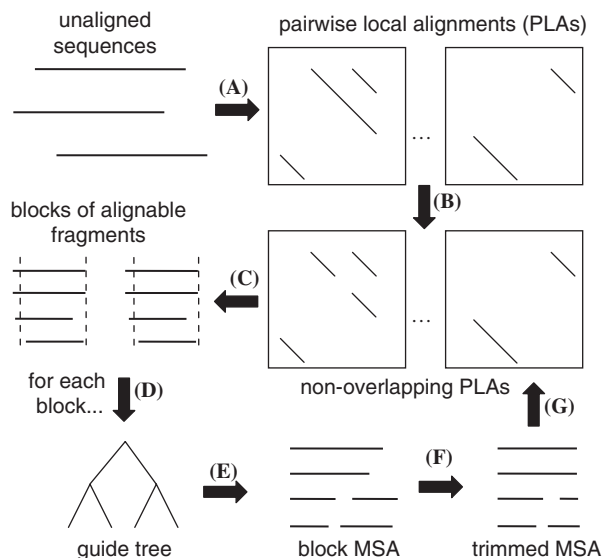


Figure 1. Overview of the ProDA algorithm. (A) Compute all-versus-all PLAs. (B) Identify possible repeats by computing non-overlapping PLAs. (C) Generate a block of possibly alignable regions from the PLAs. (D) Compute a guide tree for the block. (E) Progressively align sequences in the block and eliminate spurious alignments using heuristic filters. (F) Trim the block alignment. (G) Discard used PLAs, and repeat the alignment process until no more alignable blocks of length at least L_{\min} can be found. The end result is a set of aligned regions with length at least L_{\min} .

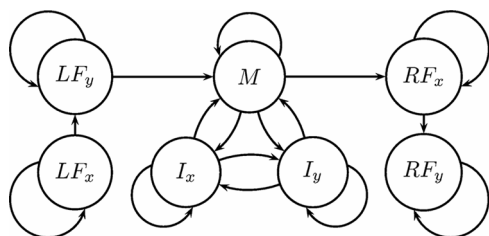


Figure 2. Pair HMM for local alignment of two sequences x and y . State M emits two letters, one from each sequence, and corresponds to the two letters being aligned together. State I_x emits a letter in sequence x that is aligned to a gap, and similarly state I_y emits a letter in sequence y that is aligned to a gap. States LF_x and LF_y emit two unaligned flanking subsequences on the left of the local alignment. Similarly, states RF_x and RF_y emit two unaligned flanking subsequences on the right of the local alignment.

with the maximum number of alignable fragments. Define the boundaries of each fragment within the block by averaging the boundary positions of PLAs corresponding to the block fragments.

Step 4: Construction of guide tree and adjustment of the block

For the block B from Step 3:

- (1) For every pair of sequence fragments $b, c \in B$ compute the posterior probabilities $P(b_i \sim c_j | b, c)$ that letters $b_i \in b$ and $c_j \in c$ are paired in an alignment generated by the pair HMM of Figure 3.
- (2) Define the expected accuracy of a pairwise alignment A between b and c to be the expected number of correctly

aligned pairs of letters, divided by the length of the shorter sequence:

$$E(\text{accuracy}(A) | b, c) = \frac{1}{\min(|b|, |c|)} \sum_{b_i \sim c_j \in A} P(b_i \sim c_j | b, c).$$

Construct a guide tree for B via hierarchical clustering, using $E(b, c) = \max_A E(\text{accuracy}(A) | b, c)$ as a measure of similarity between two fragments b and c . Stop the clustering procedure early if the similarity between two clusters to be merged drops below a threshold. Select the largest tree T built up to this point as the guide tree for B .

- (3) Retain only fragments belonging to tree T and remove all the other fragments from B .
- (4) Re-estimate the match scores $P(b_i \sim c_j | b, c)$ using the probabilistic consistency transformation, which incorporates similarity of b and c to other fragments from B into the b - c pairwise comparison.

$$P(b_i \sim c_j | b, c) \leftarrow \frac{1}{|B|} \sum_{d \in B} \sum_{d_k} P(b_i \sim d_k | b, d) P(d_k \sim c_j | d, c).$$

Step 5: Progressive alignment of the block

For the block and corresponding tree built in Step 4 progressively align fragments according to the order specified in the tree. Alignments are scored using a sum-of-pairs scoring function in which aligned residues are assigned the match quality scores $P(b_i \sim c_j | b, c)$ and gap penalties are set to zero.

Step 6: Extraction of final alignments from block alignment

Extract the longest fraction of the block alignment that begins and ends with columns containing no gaps.

Step 7: Removal of used PLAs

Remove PLAs corresponding to the fragments of the block to prevent them from contributing to subsequent blocks. If no PLAs remain, stop the algorithm. Otherwise go to Step 3.

The end result of the algorithm is a set of aligned regions with length at least L_{\min} . An example alignment generated via the ProDA algorithm is shown in Figure 3.

Algorithm details

In this section, we provide a detailed description of each step in the ProDA algorithm.

1. Pairwise local alignment. The ProDA algorithm begins by performing all-versus-all pairwise local alignments (PLAs). For each pair of sequences x and y in S , ProDA computes a set of high-scoring local alignments using an iterative procedure; in each step of the iterative procedure, a single PLA is found using either a variant of posterior decoding (described below) or the Viterbi algorithm.

In practice, computing all-versus-all local alignments is the most computationally demanding part of the algorithm. On a Pentium IV 3.6 GHz system with 2 GB memory, ProDA processes all 86 reference sets of proteins within BALiBASE reference 6 in ~ 2.5 h when using posterior decoding, and in < 1 h when using Viterbi decoding for pairwise alignment.

Pairwise alignment using posterior decoding. For each pair of input sequences x and y , ProDA models the probability distribution over all possible alignments using the pair-HMM

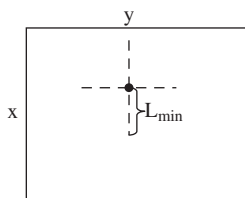


Figure 4. Marking of used positions in the alignment matrix. If a pair of letters x_i and y_j are paired in the current local alignment, they are marked (with a closed circle) to denote their exclusion from future local alignments. In addition, all letter pairs within L_{\min} residues of (x_i, y_j) in either sequence x or sequence y are also marked (with dotted lines) in order to prevent ProDA from identifying repeats of length shorter than L_{\min} .

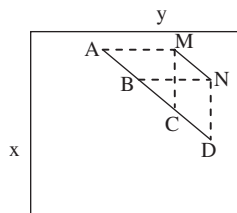


Figure 5. Breaking long local alignments that span several copies of a repeat. The two local alignments AD and MN overlap in both sequence x and y . Thus, we split AD into AB, BC and CD in order to obtain a set of non-overlapping PLAs.

where the sum is taken over all aligned residue pairs $x_i \sim y_j$ in A . ProDA computes the local alignment A^* that maximizes $q(A)$ using the Smith–Waterman algorithm (13).

- (d) If A^* is shorter than L_{\min} then stop. Otherwise, store the local alignment found and proceed to (e).
- (e) For each aligned residue pair $x_i \sim y_j$ in the local alignment A^* , we wish to ignore future local alignments from reusing the same residue pair or identifying a repeat of size shorter than L_{\min} . To accomplish this, we mark all candidate letter pairs that are within L_{\min} residues of (x_i, y_j) in either sequence x or sequence y ; i.e. set

$$Z \leftarrow Z \cup \{(x_{i+k}, y_j), (x_i, y_{j+k})\}$$

for all k such that $-L_{\min} < k < L_{\min}$ (Figure 4).

- (f) Using our new augmented set Z of disallowed residue pairs, go back to Step (b) and recompute the posterior probabilities, noting that we disallow all HMM paths which attempt to align pairs of positions in Z .

Finding all local pairwise alignments between two sequences takes time $O(nL^2)$, where L is the length of each sequence and n is the number of local alignments. For m sequences, the entire all-versus-all PLA computation takes $O(nm^2L^2)$ time. The above algorithm can be used to align a sequence against itself to find repeats by first marking the diagonal as described above. The current implementation of ProDA does not use this option, but instead finds repeats during Step 2.

Pairwise alignment using Viterbi decoding. Alternatively, a set of local alignments between sequences x and y can be computed using Viterbi decoding, by performing steps (a) through (f) above except that we now skip Step (b) and replace Step (c) with the following:

(c') Compute the local alignment A^* that maximizes the alignment probability $P(A \mid x, y, Z)$ using the Viterbi algorithm (16).

We note that there is no need to recompute the entire Viterbi dynamic programming tables in each iteration of procedure above since only a portion of the tables will be affected when marking positions to be added to Z . The Waterman–Eggert algorithm (34) uses this observation to compute the set of top-scoring Viterbi parses efficiently. If each local alignment has length $l < L$ and there are n alignments for each sequence pair, finding all local alignments between two sequences takes time $O(L^2) + O(nl^2)$. Thus,

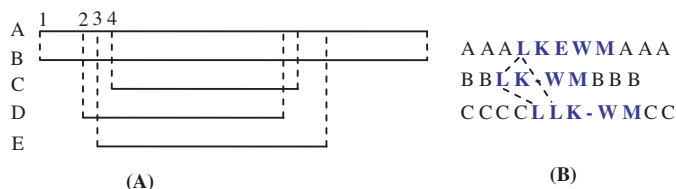


Figure 6. Challenges in determining fragment boundaries. (A) PLAs have different overlaps. (B) Different alignments are inconsistent making it hard to decide whether the third fragment should begin from the first or the second 'L'.

the local alignment step for m sequences takes time $O(m^2L^2) + O(nm^2l^2)$.

2. Inference of repeats from pairwise alignments. When two sequences share several nearby homologous repeats, a local alignment algorithm will generally find long local alignments spanning several repeat copies, along with shorter alignments containing fewer repeat copies. When two local alignments overlap in both sequences, ProDA breaks them into shorter alignments before proceeding to the subsequent steps (Figure 5).

If there are originally n local alignments of two sequences, the number of resulting non-overlapping PLAs is at most $O(n^2)$ [e.g. n copies of a perfect tandem repeat yield $n(n-1)/2$ PLAs]. For a local alignment A , computing the set of sequence x boundaries for all alignments that overlap A in sequence y takes $O(n)$ time and splitting A takes $O(l)$ time, so the total running time for repeat inference is $O(nm^2l)$.

3. Blocks of alignable sequence fragments. From an arbitrary set of PLAs, ProDA forms blocks, or sets of aligned fragments in which at least one fragment is alignable to all others either directly or via a third fragment, using a greedy iterative procedure. In each pass, ProDA selects the largest possible set of sequences to form a block. Then ProDA attempts to determine boundaries for each of the sequences in the block. This latter step can be difficult since PLAs have different lengths and boundaries (Figure 6A), and moreover, the pairwise alignments within a set of PLAs may be inconsistent with each other (Figure 6B).

ProDA uses a simple heuristic for fragment boundary determination. Recalling the definition of a block, let b denote a sequence fragment alignable to all other sequences within the current block whose boundaries we wish to determine. After filtering for outliers (see below), ProDA computes the average index of the first and last residue of sequence b in

each of the PLAs; these averaged beginning and end coordinates and their projections to the other sequences form the boundaries of the new block.

Long PLAs that contain many repeats or short PLAs that are similar to parts of real homologous fragments may cause dramatic skews in the average beginning and end index for the block with respect to sequence *b*. To prevent this from occurring, we filter out PLAs whose beginning or ending residue position in sequence *b* differs from the mean by >1 SD, and recompute average beginning and end indices for each block after this filtering has been done. For example, in Figure 6A, point 1 is an outlier so the left boundary of the block is computed by averaging points 2, 3 and 4.

4. Guide tree construction. ProDA uses the same procedure as in ProbCons to build a tree with high expected alignment reliability. In particular, given a set of sequence fragments from a block, define the similarity function $E(b,c)$ to be the expected accuracy of aligning two fragments *b* and *c*. Initially, each fragment is placed in its own cluster. Then, the two most similar clusters are merged to form a new cluster *bc*. The similarity between *bc* and any other cluster *d* is defined as $E(b,c)[E(b,d) + E(c,d)]/2$.

ProDA stops merging when the similarity between all pairs of clusters drops below some threshold (0.5 in the current implementation) or when only one cluster remains. The tree corresponding to the largest cluster formed so far is returned. All fragments not belonging to the tree are removed from the block. This early termination removes unrelated fragments that were mistakenly added to the blocks because of errors made in previous steps.

For each pair of sequences from the adjusted block, ProDA applies the probabilistic consistency transformation as used by ProbCons. As a default, ProDA uses two iterated applications of this transformation, which work well in practice (10). As with ProbCons, the tree construction and probabilistic consistency transformation steps require $O(m_b^3 c L_b)$ time, where m_b and L_b are the number of fragments and the length of each fragment of block *b*, and *c* is the average number of non-zero elements in posterior probability matrices.

5. Progressive alignment. This step is similar to the progressive alignment step of ProbCons. For each progressive alignment step, we run a profile–profile Needleman–Wunsch alignment procedure in which the score for matching a column containing n_1 non-gap letters to one with n_2 non-gap letters is computed by summing $n_1 n_2$ values from the corresponding pairwise posterior matrices. No gap penalties are used.

6. Final alignment extraction. The start and end columns of the multiple alignment from Step 5 often contain gaps as in the example shown in Figure 7. These gaps correspond to errors in the aligned fragment boundaries and should not be present in the final alignment. Thus, ProDA extracts and returns the longest aligned region whose initial and final columns contain no gaps (the region inside the rectangle in Figure 7).

7. Removing used PLAs. If two fragments belong to an alignment formed in Step 6, their aligned portions should not occur together in subsequent alignments returned by ProDA. To guarantee this, ProDA identifies all PLAs that contain

```

YAZ4_SCHPO      AGQYG-RVF-AHKFYLAARSSYAAGALREREALSAD
YIA1_YEAST      -----RIYPCHL-AVLSRAEYPKNYIE-----
YIA1_YEAST      --SNGKLFT-AHKFLLCARSET-----
008764          --VEGKLFY-AHKVLLVTASNRP-----
045398          ---K-SVP-AHKYVLAARTDF-----
074881          --KDG-ELK-AHGLFLSLRSEYPSNYITPVN-----

```

Figure 7. An alignment with gaps in the start and end columns. Block boundaries are determined by extracting the longest subalignment that begins and ends with columns not containing gaps.

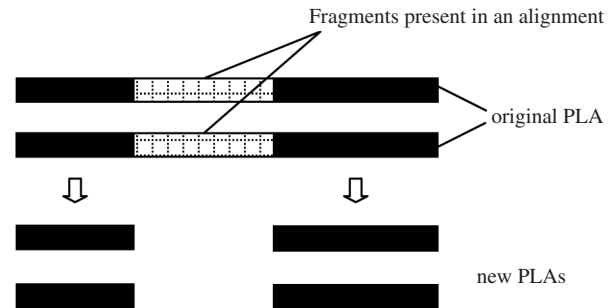


Figure 8. Adjustment of a PLA containing two fragments from a final block alignment.

pairs of fragments from the final alignment above, and removes the used portions of these PLAs from the set of candidate PLAs.

This is illustrated in Figure 8, in which the middle section of a PLA belongs to a final alignment generated in Step 5. ProDA removes this middle part, and retains the remaining left and right portions only if they are at least L_{\min} in length.

If there are no more PLAs then the algorithm stops; otherwise it goes back to Step 3.

Algorithm evaluation

For global alignments, the sum-of-pairs (SP) score is the most common measure of aligner accuracy. For alignments of proteins with rearrangements, however, using the SP score in isolation gives a poor 1D view of the aligner's overall ability to recover the domain organization of sets of related sequences. To evaluate the empirical performance of ProDA for this task, we define measures that assess the sensitivity and specificity of an aligner at the residue, domain and cluster levels.

1. Residue-level accuracy. Let **R** be a collection of reference PLAs. Similarly, let **T** be a corresponding collection of predicted PLAs generated by an alignment algorithm. To assess the residue-level accuracy, we define the following:

Residue sensitivity (**R**, **T**) =

$$\frac{\text{Number of base pairings in } \mathbf{R} \text{ that also appear in } \mathbf{T}}{\text{Number of base pairings in } \mathbf{R}}$$

Residue specificity (**R**, **T**) =

$$\frac{\text{Number of base pairings in } \mathbf{T} \text{ that also appear in } \mathbf{R}}{\text{Number of base pairings in } \mathbf{T}}$$

Here, our measure of residue sensitivity is equivalent to the SP score used for global multiple alignments.

2. Domain-level accuracy. We now characterize the ability of an aligner to recapitulate known domain boundaries (endpoint agreement) and domain positions (midpoint agreement).

Endpoint agreement. Let s and t be two (not necessarily distinct) segments. The endpoint agreement of s and t measures how well their left and right boundary positions agree. If s has boundary positions i_1 and i_2 and t has boundary positions j_1 and j_2 , then the endpoint agreement of s and t is as follows:

$$\text{Agreement}_{EP}(s, t) = \begin{cases} \frac{f(i_1, j_1) + f(i_2, j_2)}{2} & \text{if } s \text{ and } t \text{ come from the same sequence} \\ 0 & \text{otherwise} \end{cases},$$

where

$$f(i, j) = \begin{cases} 1 & \text{if } |i - j| \leq c \\ 2 - |i - j|/c & \text{if } c < |i - j| \leq 2c, \\ 0 & \text{if } |i - j| > 2c \end{cases}$$

is a function of the distance between i and j (ranging from 0 to 1). [Here, two positions are considered completely matching whenever they are within a distance c from each other, not matching if they are more than $2c$ away from each other, or partially matching otherwise; in this last case, the function $f(i, j)$ expresses the degree of matching as a real value from 0 (not matching) to 1 (completely matching).]

Given a collection of reference PLAs \mathbf{R} and a collection of predicted PLAs \mathbf{T} , we define domain endpoint sensitivity and domain endpoint specificity as follows:

$$\text{Domain sensitivity}_{EP}(\mathbf{R}, \mathbf{T}) = \frac{\sum_{\text{all domains } s \text{ in } \mathbf{R}} \max_{\text{all domains } t \text{ in } \mathbf{T}} \text{Agreement}_{EP}(s, t)}{\text{Number of domains in } \mathbf{R}},$$

$$\text{Domain specificity}_{EP}(\mathbf{R}, \mathbf{T}) = \frac{\sum_{\text{all domains } t \text{ in } \mathbf{T}} \max_{\text{all domains } s \text{ in } \mathbf{R}} \text{Agreement}_{EP}(s, t)}{\text{Number of domains in } \mathbf{T}}.$$

In the first formula, note that each of the terms in the summation is ~ 1 whenever a domain s is similar to some domain in the test collection (with respect to endpoint agreement), and is ~ 0 whenever s is not similar to any domain in the test collection. Therefore, the summation in the numerator approximates the number of \mathbf{R} domains with good representatives in \mathbf{T} , so the fraction as a whole estimates the proportion of domains in \mathbf{R} which also ‘appear’ in \mathbf{T} . Conversely, the second formula estimates the proportion of domains in \mathbf{T} which also ‘appear’ in \mathbf{R} .

Midpoint agreement. In some cases (such as low-identity alignments or overly conservative reference annotations), a good alignment algorithm may systematically overalign or

underalign annotated domains in the reference data. If this occurs in a roughly symmetric fashion, then the midpoints of the predicted segments may still match with annotations, even when their exact boundaries do not. In these cases, we can use the midpoint agreement of s and t , defined as follows:

$$\text{Agreement}_{MP}(s, t) = \begin{cases} f\left(\frac{i_1 + i_2}{2}, \frac{j_1 + j_2}{2}\right) & \text{if } s \text{ and } t \text{ come from the same sequence} \\ 0 & \text{otherwise} \end{cases}.$$

Based on this measure, we can define midpoint domain sensitivity and midpoint domain specificity by using midpoint agreement instead of endpoint agreement in the formulas above.

3. Cluster-level accuracy. Finally, we define a measure that describes how well domains are clustered into globally alignable sets. In particular, our measure assumes that the reference annotation is complete and correct. An aligner should not fail to cluster homologous sequences: segments belonging to the same reference cluster should appear in the same predicted cluster. Conversely, an aligner should not overpredict homology: segments belonging to the same predicted cluster should belong to the same reference cluster.

Let s_1 and s_2 be two distinct aligned domains s_1, s_2 in \mathbf{R} (i.e. s_1 and s_2 belong to the same PLA). If there exist distinct aligned domains t_1, t_2 in \mathbf{T} such that s_1 and t_1 overlap in at least 50% of their respective lengths, and similarly for s_2 and t_2 , then we say that s_1 and s_2 are alignable through \mathbf{T} . Then, we may define cluster sensitivity and cluster specificity as follows:

$$\text{Cluster sensitivity}(\mathbf{R}; \mathbf{T}) = \frac{\text{Number of aligned domain pairs in } \mathbf{R} \text{ that are alignable through } \mathbf{T}}{\text{Number of aligned domain pairs in } \mathbf{R}},$$

$$\text{Cluster specificity}(\mathbf{R}; \mathbf{T}) = \frac{\text{Number of aligned domain pairs in } \mathbf{T} \text{ that are alignable through } \mathbf{R}}{\text{Number of aligned domain pairs in } \mathbf{T}}.$$

Intuitively, these two measures compute the proportion of pairwise homologies in one collection of PLAs that are supported by a corresponding homology in the other collection of PLAs.

RESULTS

We performed an empirical evaluation of ProDA on existing reference alignments with repeats and rearrangements. Our experiments measured the ability of ProDA to identify correct residue pairings, find domain boundaries and group similar domains of related proteins.

Testing methodology

Most existing global and local alignment algorithms find ordered homologies, but deal with neither the occurrence of repeated/shuffled domains nor the issue of domain boundary identification. To illustrate the amount of homology sacrificed when imposing the order constraint, we evaluated the residue-level accuracy of several existing tools—the local aligner DIALIGN 2.2 (8,17,35,36), and three popular global aligners ClustalW 1.83 (37), MUSCLE 3.6 (38) and ProbCons 1.1 (10)—on reference 6 of the BALiBASE 2.0 alignment benchmark (15), a publicly available collection of 86 hand-annotated multiple alignments of protein sequences with repeats and rearrangements. (The RAlign program (33), which computes global alignments taking repeats into account, was not available for evaluation).

Reference 6 is organized into seven subsets containing sequences with different configurations of protein domains: subsets C1a and C1b consist of a few sequences with repeated domains of the same subtype; C2a, C2b and C2c, sequences with repeats of different subtypes; C3, sequences containing repeated domain and an additional non-repeated domain; and C4, sequences with shuffled domains from different protein families.

We also tested two repeat finders, RADAR (30) and TRUST (32), which were originally designed to identify homologous blocks in single sequences. To adapt these methods for generating multiple sequence homologous block sets, we used a straightforward reduction to transform our problem into an instance of repeat finding in a single sequence: given multiple input sequences, we provided the concatenation of the sequences as input to the repeat finder; any repeats identified that encompassed sequence boundaries were split, and the final set of identified repeats was filtered to eliminate homologous segments less than five residues in length.

For the task of uncovering domain organization in multiple sequence sets, the ABA program (12) is currently the only existing program that directly computes the set of homologous regions common to multiple sequences. The original implementation of ABA computes alignments for

homologous segments of one or more sequences using ClustalW and projects these down to a single consensus sequence. In order to obtain a measure of ABA's residue level accuracy, we modified the ABA script to output the intermediate ClustalW multiple alignments for homologous segments directly, and retained all segments containing at least two aligned sequences.

Finally, we tested our implementation of the ProDA algorithm with minimum alignment length $L_{\min} = 20$, using either the Viterbi or posterior decoding algorithms for computing initial pairwise alignments (Materials and Methods). In all experiments, emission probabilities for the pair HMM were adapted from the BLOSUM62 scoring matrix (39). The default transition parameters of ProDA were trained via unsupervised expectation-maximization (EM) on unaligned sequences from references 1–5 of the BALiBASE database (40). Note that we did not train on reference 6, nor did we use any reference annotations from BALiBASE.

We note that a number of the programs crashed on one or more of the alignments; in particular, on an Intel Xeon 3.2 GHz machine with 6 GB memory, ABA failed on 8 alignments, TRUST failed on 10 alignments and RADAR failed on 1 alignment. To provide a fair comparison, we excluded these alignments from our evaluation, assuming that the excluded alignments do not result in major biases in overall performance.

Residue-level accuracy

In our first set of experiments, we assessed the residue-level accuracies of all methods using our defined residue-level accuracy measures. The results for each subset of reference 6 and the entire reference are shown in Table 1.

As expected, the local aligner and three global aligners have the lowest SP scores, owing to the restriction that all elements must appear in the same order in all aligned sequences. The ABA aligner, designed specifically for these types of alignments, shows better sensitivity but in fact performs worse than the modified repeat finders. The best sensitivities are obtained by RADAR and ProDA, though both versions of

Table 1. Residue-level accuracy of aligners on BALiBASE reference 6

Score	Method	C1a (4)	C1b (6)	C2a (10)	C2b (20)	C2c (20)	C3 (5)	C4 (5)	Overall (70)
Residue sensitivity	ClustalW	0.30	0.26	0.12	0.15	0.16	0.09	0.11	0.16
	DIALIGN	0.25	0.25	0.12	0.14	0.18	0.10	0.12	0.16
	MUSCLE	0.32	0.27	0.12	0.15	0.19	0.09	0.12	0.17
	ProbCons	0.37	0.28	0.12	0.16	0.20	0.11	0.14	0.18
	ABA	0.74	0.51	0.12	0.31	0.22	0.09	0.20	0.27
	TRUST	0.38	0.42	0.26	0.22	0.31	0.31	0.19	0.28
	RADAR	0.58	0.52	0.42	0.38	0.47	0.60	0.35	0.45
	ProDA-Posterior	0.59	0.60	0.32	0.40	0.40	0.45	0.39	0.42
	ProDA-Viterbi	0.62	0.62	0.37	0.41	0.44	0.51	0.47	0.45
	Residue specificity	ClustalW	0.47	0.34	0.63	0.55	0.33	0.14	0.25
DIALIGN		0.45	0.41	0.63	0.60	0.47	0.28	0.40	0.50
MUSCLE		0.51	0.35	0.63	0.56	0.39	0.15	0.30	0.45
ProbCons		0.61	0.44	0.63	0.60	0.45	0.22	0.39	0.51
ABA		0.80	0.56	0.59	0.48	0.28	0.05	0.35	0.42
TRUST		0.57	0.56	0.65	0.62	0.59	0.56	0.54	0.60
RADAR		0.60	0.53	0.55	0.57	0.65	0.62	0.61	0.59
ProDA-Posterior		0.76	0.73	0.64	0.73	0.73	0.67	0.67	0.71
ProDA-Viterbi		0.83	0.69	0.67	0.75	0.70	0.67	0.69	0.71

Table 2. Domain-level accuracy of aligners on BALiBASE reference 6

Score	Method	C1a (4)	C1b (6)	C2a (10)	C2b (20)	C2c (20)	C3 (5)	C4 (5)	Overall (70)
Domain endpoint sensitivity	ABA	0.79	0.51	0.06	0.24	0.20	0.11	0.30	0.25
	TRUST	0.48	0.62	0.24	0.33	0.53	0.58	0.30	0.42
	RADAR	0.80	0.58	0.67	0.71	0.68	0.87	0.72	0.70
	ProDA-Viterbi	0.73	0.73	0.82	0.77	0.76	0.79	0.86	0.78
	ProDA-Posterior	0.88	0.73	0.91	0.80	0.75	0.79	0.86	0.80
Domain endpoint specificity	ABA	0.45	0.24	0.19	0.23	0.16	0.05	0.26	0.21
	TRUST	0.38	0.50	0.38	0.40	0.49	0.49	0.56	0.45
	RADAR	0.43	0.28	0.55	0.51	0.40	0.48	0.44	0.45
	ProDA-Viterbi	0.46	0.41	0.61	0.60	0.48	0.47	0.52	0.53
	ProDA-Posterior	0.60	0.41	0.70	0.63	0.53	0.50	0.58	0.58
Domain midpoint sensitivity	ABA	0.73	0.53	0.10	0.41	0.18	0.20	0.42	0.32
	TRUST	0.50	0.54	0.31	0.31	0.51	0.53	0.30	0.41
	RADAR	0.72	0.51	0.68	0.81	0.66	0.85	0.72	0.71
	ProDA-Viterbi	0.91	0.71	0.90	0.85	0.83	0.90	0.91	0.85
	ProDA-Posterior	0.92	0.75	0.93	0.86	0.84	0.95	0.87	0.86
Domain midpoint specificity	ABA	0.38	0.18	0.41	0.30	0.05	0.07	0.23	0.22
	TRUST	0.28	0.45	0.67	0.40	0.43	0.46	0.54	0.46
	RADAR	0.33	0.19	0.58	0.53	0.37	0.45	0.43	0.44
	ProDA-Viterbi	0.48	0.39	0.65	0.63	0.48	0.53	0.53	0.55
	ProDA-Posterior	0.56	0.41	0.71	0.66	0.53	0.59	0.58	0.59

ProDA significantly outperform RADAR in terms of residue-level specificity. Between the two versions of ProDA that differ in their initial pairwise alignment step, posterior decoding performs slightly worse than Viterbi decoding for residue-level accuracy.

Domain-level accuracy

Next, we measured the domain-level accuracy of TRUST, RADAR, ABA and ProDA in recovering known segment endpoints and midpoints. All scores were calculated with $c = 10$ (i.e. we only require endpoint and midpoint predictions to be correct to within 10 positions). Note that this value of c makes the endpoint measures more strict than measures used previously (14,41), where endpoints within 20 residues from the reference endpoints were counted as correct. The results are summarized in Table 2.

In these tests, ABA performs moderately well on subsets 1a and 1b, which contain only repeats of a unique domain subtype, but significantly worse on the other sets, which either contain variable subtypes or additional conserved domains. TRUST outperforms ABA, achieving comparable specificity to RADAR, but considerably lower endpoint and midpoint sensitivity. RADAR, which achieved comparable sensitivity to ProDA at the residue-level, has lower sensitivity and specificity compared to ProDA at the domain-level. Here, RADAR may be overextending domain boundaries on average, thus yielding similar residue-level sensitivity but lower accuracy in the other measures; this hypothesis is consistent with the observation that conserved segments identified by RADAR and ProDA average 52 and 33 residues in length, respectively.

The two ProDA versions stand alone in achieving both the highest sensitivity and specificity for domain endpoint and midpoint identification. ProDA's midpoint accuracies are higher than its endpoint accuracies, indicating that in many cases, ProDA correctly identifies a conserved segment but chooses different boundaries from those in the reference set, owing to overalignment or underalignment. Overall, we

conclude that ProDA correctly identifies the boundaries or midpoints of $\sim 80\text{--}85\%$ of protein domains on average.

Looking closer at alignments with low scores, we found two general types of domain configurations that proved difficult for the algorithm:

- (1) Long domains that contain weakly conserved regions. In such cases, the pairwise alignment step finds only fragments of domains. Each final alignment therefore contains conserved fragments from different domains. Here we do not miss whole domains, but we cannot find accurate boundaries. An example is the family of *Dead* domains.
- (2) Short domain repeats that are located close to each other. In such cases, the pairwise alignment step tends to over-align and finds local alignments that contain whole domains along with parts of their neighbors. The neighbors are therefore missed. Then, the algorithm outputs alignments containing fewer domains than in references but with accurate boundaries.

In both cases, errors can be traced to decisions made during the pairwise alignment step.

Cluster-level accuracy

Finally, we measured the cluster-level accuracy of the five alignment algorithms tested in the previous section. The performances of the five aligners on Reference 6 are shown in Table 3.

Here, ABA and TRUST have the least success in recovering pairwise homologous segments, each detecting $\sim 20\%$ of the aligned segment pairs in the reference annotation. RADAR and the two versions of ProDA achieve comparable sensitivities, both recovering slightly more than half of these homologies; however, ProDA is considerably more specific with respect to not declaring pairwise homologous segments that are unsupported in the reference set.

Table 3. Cluster-level accuracy of aligners on BALiBASE reference 6

Score	Method	C1a (4)	C1b (6)	C2a (10)	C2b (20)	C2c (20)	C3 (5)	C4 (5)	Overall (70)
Cluster sensitivity	ABA	0.86	0.33	0.00	0.25	0.13	0.00	0.10	0.19
	TRUST	0.39	0.28	0.11	0.12	0.31	0.27	0.16	0.21
	RADAR	0.83	0.60	0.41	0.50	0.56	0.74	0.46	0.55
	ProDA-Viterbi	0.93	0.65	0.49	0.44	0.52	0.63	0.68	0.55
	ProDA-Posterior	0.87	0.79	0.45	0.53	0.42	0.71	0.45	0.53
Cluster specificity	ABA	0.32	0.18	0.00	0.10	0.07	0.00	0.15	0.09
	TRUST	0.47	0.43	0.17	0.29	0.44	0.56	0.55	0.38
	RADAR	0.65	0.37	0.46	0.55	0.51	0.70	0.65	0.53
	ProDA-Viterbi	0.75	0.56	0.76	0.72	0.61	0.71	0.78	0.69
	ProDA-Posterior	0.74	0.60	0.76	0.77	0.59	0.78	0.71	0.70

For most of the subsets of reference 6 (except C4), the reference contains only one repeat cluster per family. Instead of finding one cluster per family, however, we observed that most aligners output several smaller clusters all belonging to the same family. This reflects the fact that domains from the same family have different degrees of conservation to each other; here, the extent of cluster fragmentation depends on the threshold used for detecting segment homology. In such a situation, we expect relatively high cluster specificities (as all domains within a predicted cluster will necessarily belong to the same cluster in the reference collection) yet low cluster sensitivities (as domains in the single reference cluster will match poorly with the split clusters in the predicted collection). This hypothesis is consistent with the data, as most methods (except ABA) achieve considerably higher cluster specificity than sensitivity.

DISCUSSION

We have described a new algorithm, ProDA, for alignment of protein sequences with repeated and shuffled domains. Given a set of sequences, ProDA first infers repeat and domain structures based on all-versus-all pairwise local alignments. It then performs progressive multiple alignment for each inferred protein block. To ensure reliability of the alignments found, ProDA uses a cascade of heuristic filters to remove spurious alignments both during alignment construction and post-processing. Unlike other multiple alignment algorithms which either ignore the problem of repeats and shuffles, or which leave the user to decide on the proper domain structure of an alignment manually, ProDA attempts to handle these issues automatically, thus presenting the user with a clear view of domain organization within proteins.

Ideally, given a set of protein sequences, an alignment tool should produce all significant homologies without imposing biologically unrealistic restrictions on how these homologies are organized. ProDA represents a significant step towards redefining the goals and expectations of such a protein multiple alignment system.

In practice, a number of challenges still remain. When the true alignment is in fact global, traditional multiple alignment methods, which do not need to decide whether observed similarity is sufficiently strong to report, tend to have an advantage. Indeed, on the globally aligned references 1–5 of BALiBASE, ProDA's average SP score of 0.73 is lower than ClustalW's average SP score, 0.86. Nevertheless, for

sequences with repeats and rearrangements, empirical tests show that ProDA recovers known protein domain structures with high accuracy while using reasonable computational resources. On reference 6 of BALiBASE, ProDA achieves significantly higher scores than other alignment programs, obtaining state-of-the-art sensitivity and substantially better specificity than previous methods.

We conclude that ProDA is a practical tool that should aid molecular biologists in tasks requiring automatic analysis of related multi-domain proteins with unknown domain architectures. Source code and executables for ProDA, along with testing scripts used to perform the evaluations, are publicly available at <http://proda.stanford.edu>.

ACKNOWLEDGEMENTS

Most of this work was done when T.M.P. was visiting Stanford University. We thank anonymous referees for helpful comments and suggestions. T.M.P. was supported by a Fulbright fellowship. CBD was supported by an NDSEG fellowship. This project was funded in part by the NSF CAREER Award. Funding to pay the Open Access publication charges for this article was provided by the NSF CAREER Award.

Conflict of interest statement. None declared.

REFERENCES

- Bateman,A., Coin,L., Durbin,R., Finn,R.D., Hollich,V., Griffiths-Jones,S., Khanna,A., Moxon,M.M., Sonnhammer,E.L., Studholme,D.J. *et al.* (2004) The Pfam protein families database. *Nucleic Acids Res.*, **32**, D138–D141.
- Phillips,A., Janies,D. and Wheeler,W. (2000) Multiple sequence alignments in phylogenetic analysis. *Mol. Phylogenet. Evol.*, **16**, 317–330.
- Rost,B. and Sander,C. (1994) Combining evolutionary information and neural networks to predict protein secondary structure. *Proteins*, **19**, 55–72.
- Livingstone,C.D. and Barton,G.J. (1993) Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. *Comput. Appl. Biosci.*, **9**, 745–756.
- Thompson,J.D., Plewniak,F. and Poch,O. (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, **27**, 2682–2690.
- Apic,G., Gough,J. and Teichmann,S.A. (2001) Domain combinations in archaeal, eubacterial, and eukaryotic proteomes. *J. Mol. Biol.*, **310**, 311–325.
- George,R.A. and Heringa,J. (2002) Protein domain identification and improved sequence similarity searching using PSI-BLAST. *Proteins*, **48**, 672–681.

8. Morgenstern, B., Frech, K., Dress, A. and Werner, T. (1998) DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, **14**, 290–294.
9. Feng, D.F. and Doolittle, R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.
10. Do, C.B., Mahabhashyam, M.S.B., Brudno, M. and Batzoglou, S. (2005) ProbCons: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.
11. Pellegrini, M., Marcotte, E.M. and Yeates, T.O. (1999) A fast algorithm for genome-wide analysis of proteins with repeated sequences. *Proteins*, **35**, 440–446.
12. Raphael, B., Zhi, D., Tang, H. and Pevzner, P. (2004) A novel method for multiple alignment of sequences with repeated and shuffled elements. *Genome Res.*, **14**, 2336–2346.
13. Smith, T.F. and Waterman, M.S. (1981) Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.
14. Wheelan, S.J., Marchler-Bauer, A. and Bryant, S.H. (2000) Domain size distributions can predict domain boundaries. *Bioinformatics*, **16**, 613–618.
15. Bahr, A., Thompson, J.D., Thiery, J.-C. and Poch, O. (2001) BALiBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations. *Nucleic Acid Res.*, **29**, 323–326.
16. Durbin, R., Eddy, S.R., Krogh, A. and Mitchison, G. (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK.
17. Morgenstern, B., Dress, A. and Werner, T. (1996) Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl Acad. Sci. USA*, **93**, 12098–12103.
18. Brocchieri, L. and Karlin, S. (1998) A symmetric-iterated method for the multiple alignment of protein sequences. *J. Mol. Biol.*, **276**, 249–264.
19. Edgar, R.C. and Sjölander, K. (2003) SATCHMO: sequence alignment and tree construction using hidden Markov models. *Bioinformatics*, **19**, 1404–1411.
20. Lee, C., Grasso, C. and Sharlow, M.F. (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics*, **18**, 452–464.
21. Bailey, T.L. and Elkan, C. (1994) Fitting a mixture model by expectation maximization to discover motifs in biopolymers. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **2**, 28–36.
22. Hertz, G.Z. and Stormo, G.D. (1999) Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics*, **15**, 563–577.
23. Lawrence, C.E., Altschul, S.F., Boguski, M.S., Liu, J.S., Neuwald, A.F. and Wootton, J.C. (1993) Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, **262**, 208–214.
24. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
25. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W. and Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.*, **25**, 3389–3402.
26. Schuler, G.D., Altschul, S.F. and Lipman, D.J. (1991) A workbench for multiple alignment construction and analysis. *Proteins*, **9**, 180–190.
27. Sonnhammer, E.L.L. and Kahn, D. (1994) Modular arrangement of proteins as inferred from analysis of homology. *Protein Sci.*, **3**, 482–492.
28. Bru, C., Courcelle, E., Carrère, S., Beausse, Y., Dalmar, S. and Kahn, D. (2005) The ProDom database of protein domain families: more emphasis on 3D. *Nucleic Acids Res.*, **33**, D212–D215.
29. Notredame, C. (2001) MoCCA: semi-automatic method for domain hunting. *Bioinformatics*, **17**, 373–374.
30. Heger, A. and Holm, L. (2000) Rapid automatic detection and alignment of repeats in protein sequences. *Proteins*, **41**, 224–237.
31. Heringa, J. and Argos, P. (1993) A method to recognize distant repeats in protein sequences. *Proteins*, **17**, 391–401.
32. Szklarczyk, R. and Heringa, J. (2004) Tracking repeats using significance and transitivity. *Bioinformatics*, **20**, 1311–1317.
33. Sammeth, M. and Heringa, J. (2006) Global multiple-sequence alignment with repeats. *Proteins*, **64**, 263–274.
34. Waterman, M.S. and Eggert, M. (1987) A new algorithm for best subsequence alignments with application to tRNA-tRNA comparisons. *J. Mol. Biol.*, **197**, 723–728.
35. Morgenstern, B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.
36. Subramanian, A.R., Weyer-Menkhoff, J., Kaufmann, M. and Morgenstern, B. (2005) DIALIGN-T: an improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics*, **6**, 66.
37. Thompson, J.D., Higgins, D.G. and Gibson, T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.
38. Edgar, R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.
39. Henikoff, S. and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10919.
40. Thompson, J.D., Plewniak, F. and Poch, O. (1999) BALiBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, **15**, 87–88.
41. Nagarajan, N. and Yona, G. (2004) Automatic prediction of protein domains from sequence information using a hybrid learning system. *Bioinformatics*, **20**, 1335–1360.
42. Sonnhammer, E.L.L., Eddy, S.R., Birney, E., Bateman, A. and Durbin, R. (1998) Pfam: multiple sequence alignments and HMM-profiles of protein domains. *Nucleic Acids Res.*, **26**, 320–322.