A comparison of scoring functions for protein sequence profile alignment

Robert C. Edgar¹, * and Kimmen Sjölander²

¹195 Roque Moraes Drive, Mill Valley, CA 94941, USA and ²Department of Bioengineering, University of California, Berkeley, CA 94720, USA

Received on May 14, 2003; revised on August 11, 2003; accepted on November 5, 2003
Advance Access publication February 12, 2004

ABSTRACT
Motivation: In recent years, several methods have been proposed for aligning two protein sequence profiles, with reported improvements in alignment accuracy and homolog discrimination versus sequence–sequence methods (e.g. BLAST) and profile–sequence methods (e.g. PSI-BLAST). Profile–profile alignment is also the iterated step in progressive multiple sequence alignment algorithms such as CLUSTALW. However, little is known about the relative performance of different profile–profile scoring functions. In this work, we evaluate the alignment accuracy of 23 different profile–profile scoring functions by comparing alignments of 488 pairs of sequences with identity ≤ 30% against structural alignments. We optimize parameters for all scoring functions on the same training set and use profiles of alignments from both PSI-BLAST and SAM-T99. Structural alignments are constructed from a consensus between the FSSP database and CE structural aligner. We compare the results with sequence–sequence and sequence–profile methods, including BLAST and PSI-BLAST.

Results: We find that profile–profile alignment gives an average improvement over our test set of typically 2–3% over profile–sequence alignment and ~40% over sequence–sequence alignment. No statistically significant difference is seen in the relative performance of most of the scoring functions tested. Significantly better results are obtained with profiles constructed from SAM-T99 alignments than from PSI-BLAST alignments.

Availability: Source code, reference alignments and more detailed results are freely available at http://phylogenomics.berkeley.edu/profilealignment/
Contact: bob@drive5.com

1 INTRODUCTION
Pairwise alignment is a fundamental tool in computational biology. An alignment of protein sequences can help identify homologous positions and regions, providing insights into the function or structure of an uncharacterized sequence by suggesting similarities to a protein that has been studied experimentally. A score or expectation value can be computed from the alignment, giving a measure of the relatedness of two sequences. This can be used to discriminate homologs from unrelated sequences and indicate the degree of functional or structural similarity that can be inferred reliably. It is well known that proteins of very low sequence similarity are sometimes related and share a common fold and function, but this similarity can be hard to detect in a direct comparison of the two primary sequences (Brenner et al., 1998).

We distinguish three classes of pairwise alignment algorithms. Sequence–sequence methods such as BLAST (Altschul et al., 1990) and FASTA (Pearson, 1990) use the two primary sequences alone. Profile–sequence methods (Tatusov et al., 1994) such as PSI-BLAST (Altschul et al., 1997) use the alignment of a query sequence to a set of similar sequences as a template for further search and alignment. PSI-BLAST uses a position-specific scoring matrix (PSSM; Gribskov et al., 1988) to summarize the information in a template alignment. SAM-T98 (Karpplus et al., 1998) is a similar method that uses a hidden Markov model (HMM; Krogh et al., 1994). PSSMs and HMMs are examples of profiles—statistical models that characterize a multiple sequence alignment. A PSI-BLAST PSSM contains estimated amino acid frequencies at each position; a SAM-T98 HMM additionally contains position-specific gap penalties. Recently, several profile–profile methods have been proposed. These construct an alignment of two profiles, from which a similarity score and pairwise alignment of the two query sequences can be derived [Pietrokovski, 1996; Lyngsø et al., 1999; Panchenko et al., 2000; Rychlewski et al., 2000; Edgar and Sjölander, 2004; Yona and Levitt, 2002; von Öhsen et al., 2003; Madera, 2003 (http://supfam.mrc-lmb.cam.ac.uk/PRC/); Panchenko, 2003; Sadreyev and Grishin, 2003]. Improvements in both alignment accuracy and homolog recognition are reported for these methods over profile–sequence and sequence–sequence methods. Profile–profile methods have been used in genome annotation and protein classification (e.g. Pawlowski et al., 1999, 2001; Henikoff et al., 2000). Profile–profile alignment is also the iterated step in progressive multiple alignment.
algorithms such as CLUSTALW (Thompson et al., 1994) and SATCHMO (Edgar and Sjölander, 2003).

Despite the promise of profile–profile methods, little is known about the relative performance of the proposed algorithms. Several factors, such as the optimization method (dynamic programming or stochastic sampling), sequence weighting, probability distribution estimation and gap scoring scheme, may be important. In this work, we focus on one key element in a profile–profile method, namely the scoring function that is applied to a candidate pair of positions. This scoring function is the profile–profile analog of the substitution matrix used in sequence–sequence methods. We create a test framework in which the scoring function varies but other design elements are maintained, allowing us to isolate any differences that are due to the position scoring function alone. To this end, we employ profile construction methods for homolog recognition, alignment, sequence weighting (both relative and against priors) and probability estimation that have been shown to perform well (see Sections 2.3 and A.2). We make no claim that these are optimal for any given application.

1.1 Profile–profile alignment algorithms
Most profile–profile methods can be viewed as variants of the well-known algorithms for pairwise sequence alignment (Needleman and Wunsch, 1970; Smith and Waterman, 1981; Gotz, 1982). A scoring function is defined as a sum over aligned pairs of profile positions plus gap penalties, and an optimal alignment is computed using dynamic programming. Within this framework, many variations are possible. One key issue is the choice of scoring functions for an aligned pair of profile positions (the position score). Given a finite alphabet \( A \) of size \(|A|\), a position score is constrained to be of the form of a symmetrical \(|A| \times |A|\) matrix, such as a BLOSUM (Henikoff and Henikoff, 1992) or PAM (Dayhoff et al., 1978) substitution matrix. A profile position, however, is a vector of \(|A|\) real values. In the absence of a theoretical model, any function that maps two such vectors to a real number can potentially be used. A plausible function will assign higher scores to vectors that are similar, but the appropriate definition of ‘similar’ is not clear, and little consensus is evident among the various proposals. A further discussion of position scores is found in the Appendix (Section A.1).

Profile–profile algorithms also differ in other ways. The amino acid alphabet is typically used, but FFAS (Rychlewski et al., 2000) adds a gap character and thus has a 21-component vector at each position. Most methods construct local alignments; however, von Ohsen et al.’s implementation is global (von Ohsen, personal communication). The LAMA algorithm (Pietrokovski, 1996) allows no gaps; most other algorithms apply affine gap penalties. COACH (Edgar and Sjölander, 2004) and PRC (Madera, 2003) use position-specific gap penalties derived from profile HMMs. Some methods are not formulated as a dynamic programming score optimization. Lyngsø et al. (1999) align two HMMs by considering co-emission probabilities, and Panchenko (2003) uses Gibbs sampling.

In this study, we construct local alignments of two profiles by dynamic programming optimization of a sum over position scores plus affine gap penalties. We believe that local alignment is more sensitive to the form of the position score than global alignment, due to the absence of length constraints.

2 METHODS
2.1 Reference alignments
We assessed alignment accuracy by comparison with structural alignments, focusing on alignments between pairs of low sequence identity (≤30%). As there can be significant ambiguities in the sequence alignment implied by a superposition of distantly related structures (Cline, 2000), we chose pairs of structures of high similarity, and restricted sequence alignments to regions where two independent structural aligners agreed. This was done as follows. We selected pairs of sequences from the FSSP database (Holm and Sander, 1996) having ≤30% identity, z-score ≥15, RMSD ≤2.5 Å and an alignment length of ≥50 positions. To reduce redundancy, these pairs were filtered so that no two sequences aligned to a common third sequence had >30% identity. Structures for the remaining pairs were obtained from the Protein Data Bank (Berman et al., 2000) and aligned using the CE aligner (Shindyalov and Bourne, 1998). The consensus of the CE and FSSP alignments, defined as the set of aligned residue pairs on which both agreed, was then extracted, and alignments shorter than 50 positions were eliminated. This produced a total of 588 alignments, ranging from 61 to 675 positions, with an average length of 201. Sequence identities ranged from 9.6% to 30%, with an average of 21.8%.

2.2 Alignment accuracy scoring
We used three quality scores for comparing a test alignment with a reference alignment. SP is the number of correctly aligned pairs in the test alignment, \( t_s \), divided by the length of the reference alignment. This score has been used, e.g. by Thompson et al. (1999), who call it SP, and by Sauder et al. (2000), who refer to it as \( f_D \), the developer’s score. PS (reverse sum-of-pairs) is \( t_c \) divided by the length of the test alignment; this is Sauder et al.’s \( f_M \), the modeler’s score. Each of these scores is useful in some applications but also has drawbacks. SP does not penalize over-alignment (i.e. aligning residue pairs that are not structurally alignable); PS does not penalize under-alignment. Neither gives credit for regions in the test alignment that are shifted by one or a few positions relative to the reference alignment; however, such regions may still be successfully used in homology modeling and may even be more ‘correct’ when probable homology is considered rather than atom coordinates alone. Cline et al. (2002) have proposed a score that is designed to address these issues; we
call it the Cline score (CS). It penalizes both over- and under-
alignment, and gives positive, although reduced, scores for
positions with small shifts. CS has a parameter \( \varepsilon \) that controls
the range of shifts that get positive scores; following Cline
et al. (2002), we set \( \varepsilon = 0.2 \). All three scores have a max-
imum value of 1 in the case of perfect agreement. SP and PS
have a minimum of zero when no pairs are correctly aligned;
CS can achieve negative values when there are many large
shifts.

2.3 Profile construction
Given a sequence (the seed), construction of a profile involves
several steps, including identification of homologs, aligning
those homologs to the seed (creating a multiple alignment that
we call a template), determination of sequence weights and
finally estimation of amino acid probability distributions in
each position. We used two different methods for homolog
identification and alignment: PSI-BLAST and SAM-T99, an
updated version of the SAM-T98 algorithm. We performed
parameter optimization and alignment accuracy assessment
separately for profiles produced using template alignments
from each method. We generated our own profiles rather
than using native PSSM or HMM profiles produced by these
methods. This allows us to add information not found in the
native profiles, such as the observed amino acid frequencies
before pseudo-counts are added and eliminates differences
in sequence weighting and amino acid probability estimation
used by PSI-BLAST and SAM-T99. We used the NCBI
non-redundant protein sequence database (Pruitt et al.,
2003) as our search database. As both PSI-BLAST and SAM-T99
are known to introduce false-positive hits, some alignments
will contain one or more unrelated sequences. Some investi-
gators (e.g. Yona and Levitt, 2002) have built profiles by
restricting the search database to sequences known to belong
to the same family. This avoids false positives but cannot
be applied when the seed is experimentally uncharacterized.
Further details of our profile construction are given in the
Appendix (Section A.2).

2.4 Position scoring functions
We tested 23 position-scoring functions, as defined in the
Appendix (Section A.1).

2.5 Gap and center parameters
We apply affine gap penalties to all position-scoring functions.
Denoting the gap open penalty by \( g \) and gap extension penalty
by \( e \), a gap of length \( \lambda \) is scored as \(-g - (\lambda - 1)e\). A local
alignment scoring function must produce both positive and
negative scores; otherwise the alignment will be empty (if all
scores are negative) or will be de facto global (if all scores
are positive). Scoring functions that are based on log-odds
scores, such as \( al \), \( la \) and \( coach \), are designed to do this. Other
functions, e.g. \( fdotf \) and \( yl \), are positive-definite and must
therefore be modified before they can produce local align-
ments. The simplest solution is to subtract a constant value
that we call the center and denote by \( c \). Adjusting the center
tends to change the length of a local alignment. We introduce
center parameter for all functions, including those that have
a log-odds form.

2.6 Parameter optimization
In order to achieve parity between the tested functions, we
must optimize gap and center parameters for local align-
ment. A single quality score must be selected for this
purpose. Choosing SP or PS would encourage over- or under-
alignment, respectively. We believe that the CS gives the best
single indication of accuracy and chose to optimize para-
eters based on this measure. We selected 100 from the
588 reference alignments at random for use as a training
set. Details of the optimization procedure are given in the
Appendix (Section A.3).

3 RESULTS
Some selected results are shown in Table 1. Complete results
and other supplementary materials are available from our Web
site (see Abstract section). Eliminating the three worst per-
formers that score no better than PSI-BLAST (\( yldf \), \( euclidf \)
and \( ref \)). CS scores on the complete test set using SAM-T99
templates (T99 column in Table 1) range from 0.832 (\( pdotp \))
to 0.810 (\( euclidp \)), a difference of 2.7%. A rank comparison
of \( pdotp \) against \( euclidp \) using the Friedman test shows that
this difference is not statistically significant: \( pdotp \) is better in
245 of the 488 test pairs, \( euclidp \) is better in 236 cases
and there are seven ties—a difference consistent with chance
variations. This is confirmed by observing the lack of cor-
relation in the rankings of the functions according to results
using SAM-T99 templates (T99 in Table 1) and PSI-BLAST
templates (PB).

Comparing T99 results with PB shows consistently bet-
ter results using SAM-T99. This is statistically significant
(\( p < 10^{-5} \)), with SAM-T99 giving better results in
7905 cases, PSI-BLAST better in 5144 and 475 ties. We
believe this is due to the fact that SAM-T99 included, on
average, 4.6 times as many sequences as PSI-BLAST, pre-
sumably identifying many more homologs, though possibly
at the expense of including more false positive hits than
PSI-BLAST.

To compare profile–profile methods against PSI-BLAST,
we initialized PSI-BLAST from one template alignment
and used the other seed sequence as the search database.
This gave the results shown in the \( pb\) row in Table 1. A
small, but significant, difference is found for some of the
higher scoring profile–profile functions. For example, \( pdotp \)
is better than PSI-BLAST in 310 cases and worse in 159,
with 19 ties, making \( pdotp \) superior with \( p < 0.0001 \).
One should bear in mind that our profile–profile methods


### Table 1. Here we show selected results

<table>
<thead>
<tr>
<th>SF</th>
<th>T99</th>
<th>PB</th>
<th>(\leq 15%)id</th>
<th>(\leq 15%)seq</th>
<th>SP</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>fdotp</td>
<td>0.832</td>
<td>0.805</td>
<td>0.697</td>
<td>0.737</td>
<td>0.829</td>
<td>0.806</td>
</tr>
<tr>
<td>rankp</td>
<td>0.830</td>
<td>0.807</td>
<td>0.706</td>
<td>0.717</td>
<td>0.835</td>
<td>0.794</td>
</tr>
<tr>
<td>prc</td>
<td>0.829</td>
<td>0.801</td>
<td>0.693</td>
<td>0.796</td>
<td>0.834</td>
<td>0.794</td>
</tr>
<tr>
<td>coach</td>
<td>0.829</td>
<td>0.806</td>
<td>0.697</td>
<td>0.737</td>
<td>0.830</td>
<td>0.797</td>
</tr>
<tr>
<td>corelp</td>
<td>0.829</td>
<td>0.806</td>
<td>0.702</td>
<td>0.749</td>
<td>0.835</td>
<td>0.794</td>
</tr>
<tr>
<td>corelf</td>
<td>0.829</td>
<td>0.806</td>
<td>0.709</td>
<td>0.710</td>
<td>0.831</td>
<td>0.795</td>
</tr>
<tr>
<td>yl*</td>
<td>0.829</td>
<td>0.809</td>
<td>0.668</td>
<td>0.673</td>
<td>0.834</td>
<td>0.794</td>
</tr>
<tr>
<td>yl</td>
<td>0.828</td>
<td>0.807</td>
<td>0.699</td>
<td>0.724</td>
<td>0.834</td>
<td>0.792</td>
</tr>
<tr>
<td>rankf</td>
<td>0.827</td>
<td>0.809</td>
<td>0.703</td>
<td>0.566</td>
<td>0.830</td>
<td>0.797</td>
</tr>
<tr>
<td>fDoEF</td>
<td>0.823</td>
<td>0.801</td>
<td>0.696</td>
<td>0.652</td>
<td>0.827</td>
<td>0.790</td>
</tr>
<tr>
<td>yld</td>
<td>0.822</td>
<td>0.801</td>
<td>0.692</td>
<td>0.702</td>
<td>0.827</td>
<td>0.787</td>
</tr>
<tr>
<td>re</td>
<td>0.822</td>
<td>0.798</td>
<td>0.696</td>
<td>0.783</td>
<td>0.827</td>
<td>0.787</td>
</tr>
<tr>
<td>mdotm</td>
<td>0.820</td>
<td>0.809</td>
<td>0.694</td>
<td>0.627</td>
<td>0.819</td>
<td>0.800</td>
</tr>
<tr>
<td>mdotp</td>
<td>0.820</td>
<td>0.801</td>
<td>0.687</td>
<td>0.698</td>
<td>0.826</td>
<td>0.785</td>
</tr>
<tr>
<td>fdotp</td>
<td>0.820</td>
<td>0.799</td>
<td>0.682</td>
<td>0.710</td>
<td>0.813</td>
<td>0.803</td>
</tr>
<tr>
<td>ylf</td>
<td>0.819</td>
<td>0.790</td>
<td>0.685</td>
<td>0.339</td>
<td>0.824</td>
<td>0.790</td>
</tr>
<tr>
<td>ali</td>
<td>0.819</td>
<td>0.801</td>
<td>0.671</td>
<td>0.684</td>
<td>0.825</td>
<td>0.785</td>
</tr>
<tr>
<td>al</td>
<td>0.814</td>
<td>0.800</td>
<td>0.677</td>
<td>0.717</td>
<td>0.817</td>
<td>0.784</td>
</tr>
<tr>
<td>lal</td>
<td>0.812</td>
<td>0.791</td>
<td>0.682</td>
<td>0.620</td>
<td>0.805</td>
<td>0.795</td>
</tr>
<tr>
<td>la</td>
<td>0.811</td>
<td>0.795</td>
<td>0.692</td>
<td>0.670</td>
<td>0.802</td>
<td>0.798</td>
</tr>
<tr>
<td>euclidp</td>
<td>0.810</td>
<td>0.786</td>
<td>0.666</td>
<td>0.688</td>
<td>0.815</td>
<td>0.777</td>
</tr>
<tr>
<td>pb</td>
<td>0.802</td>
<td>0.758</td>
<td>0.568</td>
<td>0.544</td>
<td>0.733</td>
<td>0.768</td>
</tr>
<tr>
<td>yldf</td>
<td>0.802</td>
<td>0.791</td>
<td>0.668</td>
<td>0.522</td>
<td>0.808</td>
<td>0.772</td>
</tr>
<tr>
<td>euclidf</td>
<td>0.788</td>
<td>0.771</td>
<td>0.624</td>
<td>0.306</td>
<td>0.793</td>
<td>0.765</td>
</tr>
<tr>
<td>ref</td>
<td>0.766</td>
<td>0.681</td>
<td>0.539</td>
<td>0.207</td>
<td>0.770</td>
<td>0.738</td>
</tr>
<tr>
<td>la*</td>
<td>0.760</td>
<td>0.769</td>
<td>0.598</td>
<td>0.595</td>
<td>0.742</td>
<td>0.788</td>
</tr>
<tr>
<td>blast</td>
<td>0.592</td>
<td>0.592</td>
<td>0.191</td>
<td>0.493</td>
<td>0.536</td>
<td>0.678</td>
</tr>
</tbody>
</table>

SF is the position-scoring function (Section A.1). SP and PS are the sum-of-correct-pairs and reverse SP accuracy scores, respectively, computed for profile–profile alignment using SAM-T99 templates. All other columns show CS scores. T99 is for profile–profile alignment using SAM-T99 templates, PB using PSI-BLAST templates. The \(\leq 15\%)id column shows scores for the subset of 83 pairs with \(\leq 15\%\) pairwise identity. The \(\leq 15\%)seq column shows scores for the 17 pairs in which one or both templates had \(\leq 15\) sequences when made non-redundant at 80% identity. Rows are sorted in decreasing order of the T99 column. The scoring function _pb_ is PSI-BLAST, as initialized from a template alignment of one seed sequence; _blast_ is BLAST on the seed sequences alone. Scoring functions _yl_ and _la_ have the authors’ recommended parameters rather than our optimized parameters. The slight improvement in T99 score of \(yl^*\) and \(la^*\), respectively, indicates that our optimization on the training set produced marginally inferior parameters for the test set.

were optimized on a training set of low sequence identity and had the advantage of the center parameter which can tune the alignment length. In order to make a fair comparison of profile–profile versus profile–sequence methods, the same procedures should be applied. We therefore optimized gap and shift parameters for profile–sequence alignment with two scoring functions: _al_, a BLOSUM62 score that reduces to that used in BLAST and PSI-BLAST when there is exactly one sequence in each profile, and _pdotp_. For this experiment, a profile was constructed from one seed sequence alone, and a profile–profile alignment made with a profile built from the SAM-T99 template for the other seed. Results are shown in Table 2. We see an improvement of 2.2% in the average CS score for profile–profile versus profile–sequence alignment using _al_, and 3.2% using _pdotp_—small but statistically significant differences. Similar results were obtained from other position score function tests performed without re-optimizing parameters.

For two of the tested scoring functions, _yl_ and _la_, the authors who proposed these functions report their preferred parameters. We tested these functions using the authors’ parameters in addition to our own and denote those functions by _yl^*_ and _la^*_, respectively. We see from Table 1 that the authors’ parameters for _yl_ give very similar results, but in the case of _la_ we see a significant improvement from using optimized parameters.

### Table 2. Comparison of profile–profile and profile–sequence alignment for two selected scoring functions, _al_ and _pdotp_

<table>
<thead>
<tr>
<th></th>
<th><em>al</em></th>
<th><em>pdotp</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>CS (prof–prof)</td>
<td>0.814</td>
<td>0.832</td>
</tr>
<tr>
<td>CS (prof–seq)</td>
<td>0.796</td>
<td>0.806</td>
</tr>
<tr>
<td>CS (prof–seq*)</td>
<td>0.791</td>
<td>0.797</td>
</tr>
</tbody>
</table>

CS (prof–prof) is the average Cline quality score (CS) for the complete test set using profile–profile alignment. CS (prof–seq) is the score for profile–sequence alignment. To investigate the change due to re-optimizing shift and gap parameters for profile–profile alignment, we tried using profile–sequence alignment with parameters optimized for profile–profile scoring, with the results shown as CS (prof–seq*). A small but not statistically significant, improvement is seen with the re-optimized parameters. Prof–prof better indicates the number of test pairs for which profile–profile alignment gave a higher CS. Prof–seq better the number where profile–sequence alignment gave a higher CS, and Ties the number for both gave the same CS. The _p_-value is the statistical significance of the result that profile–profile alignment, on average, gives more accurate results.

### 4 DISCUSSION

We assessed the alignment accuracy of 23 different profile–profile scoring functions on a test set of 488 alignments of pairs of protein structures of low identity but high structural similarity. We found that most of these functions performed slightly better than profile–sequence methods (2–3% improvement), and substantially better than sequence–sequence methods (40% improvement). No statistically significant difference was detectable between most of the functions on the full test set or on subsets of exceptionally low identity or where only a few similar sequences were identified for inclusion in the profile. This improvement in accuracy is comparable with those seen by Yona and Levitt (2002) and Sadreyev and Grishin (2003). This modest improvement in alignment accuracy over profile–sequence methods

\(^1\)Gap parameters in (Yona and Levitt, 2002) are misquoted (Golan Yona, personal communication); we use the corrected values.

\(^2\)Values were corrected for 15% pairwise identity.
may be important in some applications, such as homology modeling, and may correlate with greater improvements in homolog recognition performance, as shown by Yona and Levitt (2002) and Sadreyev and Grishin (2003). However, profile–profile methods have significantly higher computational costs.

We are aware of only one previous study that attempted to compare profile–profile scoring functions (Pietrokovski, 1996), which found a function similar to what we call correff to perform best. However, no center parameter was introduced, and so the results are not directly comparable. Sadreyev and Grishin (2003) show a slight improvement in the alignment accuracy of their profile–profile method, COMPASS, over Yona and Levitt’s (2002) prof_sim program. It is not clear whether this performance difference is due to the position-scoring function or to other factors.

The tested functions vary widely in degree of theoretical sophistication and cost in terms of execution time. The simplest, dot products such as fdotf, require only 20 multiplications and 20 additions. A more sophisticated function, yl, is motivated by information theory and requires the evaluation of 80 logarithms at each position. Our results suggest that the expense of computing the more complex functions may have no significant benefit for the accuracy of the alignment.

We found that profiles constructed from SAM-T99 alignments gave significantly better results than profiles constructed from PSI-BLAST alignments, giving a center typical average improvement of 2–3%, comparable with the improvement of profile–profile over profile–sequence methods. On average, SAM-T99 alignments contained 4.6 times as many sequences as PSI-BLAST, suggesting that more homologs were successfully identified, though possibly at the expense of including more false positive hits.

We chose to study alignment accuracy due to its intrinsic importance to applications such as homology modeling and critical residue identification and also because of its computational tractability. Alignment accuracy can be meaningfully assessed on a single alignment, whereas homolog discrimination has an intrinsic trade-off between coverage and error and must therefore be evaluated on a large number of pairs for a range of thresholds (Brenner et al., 1998). Furthermore, a potentially good method may be handicapped by a sub-optimal computation of a p-value or e-value; alignment accuracy is independent of this step.

Our results show that alignment accuracy, as assessed on our reference data, is not very sensitive to the functional form of the position score and therefore provides little guidance as to which scores will perform well as a similarity measure. Yona and Levitt (2002) report a substantial improvement in homolog discrimination from their scoring function (yl) over PSI-BLAST, despite showing (in agreement with our own findings) only a small improvement in alignment accuracy. We therefore plan to make a systematic comparison of the homolog discrimination performance of profile–profile scoring functions.

ACKNOWLEDGEMENTS

The authors are grateful to Steven Brenner, Melissa Cline, Kevin Karplus, Martin Madera, Niklas von Öhse and Golan Yona for helpful discussions.

REFERENCES


A APPENDIX

A.1 Position-scoring functions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p^0_a$</td>
<td>Background probability of amino acid $a$</td>
<td></td>
</tr>
<tr>
<td>$B_{ab}$</td>
<td>BLOSUM62 score for $a \leftrightarrow b$</td>
<td></td>
</tr>
<tr>
<td>$f_{xa}$</td>
<td>Observed (sequence weighted) frequency of $a$ at position $k$</td>
<td></td>
</tr>
<tr>
<td>$p_{xa}$</td>
<td>Estimated probability of $a$ at position $k$</td>
<td></td>
</tr>
<tr>
<td>$M_{xa}$</td>
<td>HMM match score of $a$ at position $k$</td>
<td></td>
</tr>
<tr>
<td>$x \cdot y$</td>
<td>Dot product of $x$ and $y$</td>
<td>$\sum a x_a y_a / (x_a ^2 + y_a ^2)$</td>
</tr>
<tr>
<td>$\text{Avg}(x, y)$</td>
<td>Averaged vector</td>
<td>$\sum a x_a y_a / (x_a ^2 + y_a ^2)$</td>
</tr>
<tr>
<td>$D_{KL}(x, y)$</td>
<td>Kulback–Leibler divergence</td>
<td>$\sum a x_a \log_2 x_a / y_a$</td>
</tr>
<tr>
<td>$H^S(x, y)$</td>
<td>Symmetrized entropy</td>
<td>$D_{KL}(x, y) + D_{KL}(y, x) / 2$</td>
</tr>
<tr>
<td>$D^S(x, y)$</td>
<td>Jensen–Shannon divergence</td>
<td>$D_{KL}(x, Avg(x, y)) + D_{KL}(y, Avg(x, y)) / 2$</td>
</tr>
<tr>
<td>$d(x, y)$</td>
<td>Euclidean distance</td>
<td>$\sqrt{\sum a (x_a - y_a)^2}$</td>
</tr>
<tr>
<td>$\langle x \cdot x \rangle$</td>
<td>Mean of $x$</td>
<td>$\sum a x_a^2 /</td>
</tr>
<tr>
<td>$R(x, y)$</td>
<td>Pearson correlation</td>
<td>$\frac{\sum a (x_a - \langle x \cdot x \rangle)(y_a - \langle y \cdot y \rangle)}{\sqrt{(\sum a (x_a - \langle x \cdot x \rangle)^2)(\sum a (y_a - \langle y \cdot y \rangle)^2)}}$</td>
</tr>
</tbody>
</table>

where $\sigma_n(x)$ = 1 if $x_a$ is smallest to $\sigma_n(x) = |x|$ if $x_a$ is largest; ties defined so that $\sum \sigma_n(x)$ remains constant.
We define the following scoring functions. Subscripts 1 and 2 refer to the two profile positions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>al</td>
<td>$\sum_a \sum_b f_{ab} R_{ab}$</td>
</tr>
<tr>
<td>ali</td>
<td>$\sum_a \sum_b f_{ab} [R_{ab} + 0.5]$</td>
</tr>
<tr>
<td>correlf</td>
<td>$R(f_1, f_2)$</td>
</tr>
<tr>
<td>correlp</td>
<td>$R(p_1, p_2)$</td>
</tr>
<tr>
<td>euclidf</td>
<td>$d(f_1, f_2)$</td>
</tr>
<tr>
<td>euclidp</td>
<td>$d(p_1, p_2)$</td>
</tr>
<tr>
<td>fdotf</td>
<td>$f_1 \cdot f_2$</td>
</tr>
<tr>
<td>fdotp</td>
<td>$f_1 \cdot p_2$</td>
</tr>
<tr>
<td>la</td>
<td>$\log_2 \sum_a \sum_b f_{ab} 2^{R_{ab}}$</td>
</tr>
<tr>
<td>lai</td>
<td>$\log_2 \sum_a \sum_b f_{ab} 2^{(R_{ab} + 0.5)}$</td>
</tr>
<tr>
<td>coach</td>
<td>$M_1 \cdot f_2$</td>
</tr>
<tr>
<td>mdotf</td>
<td>$M_1 \cdot p_2$</td>
</tr>
<tr>
<td>mdotp</td>
<td>$M_1 \cdot p_2$</td>
</tr>
<tr>
<td>prc</td>
<td>$\log_2 \sum_a p_{la} p_{la} / p_a^2$</td>
</tr>
<tr>
<td>rankf</td>
<td>$R[\sigma(f_1), \sigma(f_2)]$</td>
</tr>
<tr>
<td>rankp</td>
<td>$R[\sigma(p_1), \sigma(p_2)]$</td>
</tr>
<tr>
<td>re</td>
<td>$H^p(p_1, p_2)$</td>
</tr>
<tr>
<td>refl</td>
<td>$H^p(f_1, f_2)$</td>
</tr>
<tr>
<td>yl</td>
<td>$[1 - D^B(p_1, p_2)][1 + D^B(Avg(p_1, p_2), p^0)]$</td>
</tr>
<tr>
<td>yld</td>
<td>$1 - D^B(p_1, p_2)$</td>
</tr>
<tr>
<td>ylfd</td>
<td>$1 - D^B(f_1, f_2)$</td>
</tr>
<tr>
<td>ylf</td>
<td>$[1 - D^B(f_1, f_2)][1 + D^B(Avg(f_1, f_2), p^0)]$</td>
</tr>
</tbody>
</table>

The $al$ function is the sum of pairs (average-log) function as found, e.g. in CLUSTALW (Thompson et al., 1994). The $la$ function is the log-average score (von Öhsen et al., 2003). The correlf function is used in the LAMA algorithm (Pietrokovski, 1996). As with other functions, we try variants using both the observed frequencies (correlf) and estimated probabilities (correlp) derived by adding pseudo-counts computed from the Dirichlet mixture. The function fdotf is similar to the one used in the FFAS server (Rychlewski et al., 2000). coach is the COACH position score (Edgar, 2002); we include the other dot products for completeness. The rankf and euclidf functions were considered but rejected by Pietrokovski (1996) on the grounds that they were not suitable for local alignment; we address this by introducing the center parameter. The prc function is the PRC position score (Madera, 2003). The symmetrized relative entropy (re) is used in the BETE phylogenetic clustering algorithm (Sjölander, 1998). The yl function is Yona and Levitt’s (2002); the yld function contains their divergence term only; it was not clear to us whether the significance term is essential and we therefore wished to investigate the consequences of removing it. We test the possible impact of integer rounding, which preserves only one or two significant figures, in the ali and lai functions. Note that there are significant differences in the cost of computing these functions. For example, dot products perform 20 multiplications and 20 additions, and the yl function computes 80 logarithms. Note also that fdotp, coach and mdotf are asymmetrical under an exchange of the two profiles. The COMPASS scoring function (Sadreyev and Grishin, 2003) is not included here as this work was largely completed prior to its publication.

A.2 Profile construction

Default parameters were used for SAM-T99, as found in the target99 script in SAM version 3.3.1 (Hughey and Krogh, 1996). For PSI-BLAST, we used blastpgp version 2.2.5 with options -h5 -e0.1, following the advice of the author of the PSI-BLAST checkpointing code (Alejandro Schaffer, personal communication). In the case of PSI-BLAST, alignments produced by the final iteration were used, discarding any alignments from earlier iterations. Sequence weighting was done using Henikoff weights (Henikoff and Henikoff, 1994), incorporating the same modifications employed by PSI-BLAST: gaps are treated as a 21st letter, and columns that are perfectly conserved are excluded from the calculation. Probabilities were estimated using Dirichlet mixture parameters from Sjölander et al. (1996). The total sequence weight relative to the priors (the estimated number of distinct observed sequences, corrected for over-represented subfamilies) was derived using a method from the SAM package (Kevin Karplus, personal communication), as follows. We define the number of bits saved relative to the background as

$$b = \frac{1}{M} \sum_{ka} \sum_{ka} p_{ka} \log_2 \left( \frac{p_{ka}}{p_a} \right).$$

Here, $M$ is the number of HMM nodes, $k$ is the profile position, $a$ is the amino acid type, $p_{ka}$ is the estimated probability of $a$ in the $k$-th position and $p_a^0$ is the approximation to the background probability obtained by applying the Dirichlet regularizer to a vector of zero counts. We iteratively adjust the total sequence weight until $b$ converges on a target value, which in our case we choose to be 0.5, following the default in SAM.

A.3 Parameter optimization

The goal of parameter optimization is to maximize the average CS quality score for a given position score over the 100 training alignments as a function of center and gap parameters; we call this function $Q(c, g, e)$. This is a challenging problem. Note that a sufficiently small change in any parameter will leave all alignments unchanged; hence $Q$ is a discontinuous function whose partial derivatives are zero almost everywhere. In addition, a single value of $Q$ is expensive to compute, and experience shows that local maxima are commonly found. We experimented with several fully automated optimization methods, but found none to be satisfactory, and settled on the following procedure. An iterated step defines a uniform grid of $5^3$ points in the three-dimensional parameter space and computes the value of $Q$ at each point. An approximation to a local maximum is identified as an interior point in the grid whose six neighbors all have lower values. An
estimate is made of the coordinates of the maximum and a grid of reduced size evaluated with this point at its center. If no local maximum is found, the grid is enlarged and/or moved until one is found. If multiple maxima are found, each is explored. This process is repeated until convergence, with manual inspection at each step in an attempt to identify and avoid local maxima. This human intervention is unfortunate as it introduces a subjective element (and is also somewhat tedious); however, attempts to automate this procedure fully would sometimes lead to convergence on a local maximum. Even with human intervention, we saw no definitive way to avoid local maxima—as $Q$ is a step function derived from the ratios of integer values, there can be a hidden maximum between two neighboring points in the grid that have equal values of $Q$ and hence appear to have fully converged. Convergence in the value of $Q$ was typically achieved to within $\sim 0.1\%$, which roughly corresponds to shifting two residue pairs by one position over the whole test set and is therefore close to the smallest possible change in $Q$. Optimization was performed separately for each profile position-scoring function and for profiles derived from each search method (PSI-BLAST and SAM-T99). We do not claim that the parameters obtained in this fashion are necessarily optimal for any particular application; our primary aim, as elsewhere in this work, is to achieve parity between the scoring functions so that the results are directly comparable.