

# Local homology recognition and distance measures in linear time using compressed amino acid alphabets

Robert C. Edgar\*

195 Roque Moraes Drive, Mill Valley, CA 94941, USA

Received October 23, 2003; Revised November 12, 2003; Accepted November 25, 2003

## ABSTRACT

**Methods for discovery of local similarities and estimation of evolutionary distance by identifying  $k$ -mers (contiguous subsequences of length  $k$ ) common to two sequences are described. Given unaligned sequences of length  $L$ , these methods have  $O(L)$  time complexity. The ability of compressed amino acid alphabets to extend these techniques to distantly related proteins was investigated. The performance of these algorithms was evaluated for different alphabets and choices of  $k$  using a test set of 1848 pairs of structurally alignable sequences selected from the FSSP database. Distance measures derived from  $k$ -mer counting were found to correlate well with percentage identity derived from sequence alignments. Compressed alphabets were seen to improve performance in local similarity discovery, but no evidence was found of improvements when applied to distance estimates. The performance of our local similarity discovery method was compared with the fast Fourier transform (FFT) used in MAFFT, which has  $O(L \log L)$  time complexity. The method for achieving comparable coverage to FFT is revealed here, and is more than an order of magnitude faster. We suggest using  $k$ -mer distance for fast, approximate phylogenetic tree construction, and show that a speed improvement of more than three orders of magnitude can be achieved relative to standard distance methods, which require alignments.**

## INTRODUCTION

Biological databases are growing exponentially, and algorithms that minimize processor time and memory requirements are becoming increasingly important. Many of the commonly used sequence comparison algorithms are based on pairwise alignments. Database search methods such as BLAST (1) attempt to discriminate related from unrelated sequences by computing expectation values from local alignments. Clustering algorithms such as UPGMA (2) and neighbor-joining (3) typically use a distance measure that is a

function of the pairwise identity computed from an alignment. Progressive multiple alignment algorithms such as CLUSTAL\_W (4) first perform clustering to create a binary tree and then construct one pairwise alignment of two profiles for each internal node of the tree. Pairwise alignments are generally constructed using dynamic programming algorithms that have time complexity  $O(L^2)$  in the sequence length  $L$  (reviewed in 5). In this study, we combine two techniques,  $k$ -mer lookup and compressed alphabets, to achieve reductions in the time and space complexity of pairwise alignment. We describe how local similarities between two sequences (one or both of which may be a profile) can be identified with high reliability in  $O(L)$  time, and show how this can improve alignment speed with negligible reduction in accuracy. We also explore a method that estimates the pairwise identity of two unaligned sequences in  $O(L)$  time.

## Compressed alphabets

A compressed alphabet 'C' of size  $N$  is a partition of the standard 20-letter amino acid alphabet 'A' into  $N$  disjoint subsets (classes) containing similar amino acids. Several methods for constructing such alphabets have been proposed. Li *et al.* (6) created two sets of alphabets by seeking to maximize a similarity score derived from the BLOSUM62 matrix (7). A heuristic search procedure inspired by the Monte Carlo algorithm was used, yielding two sets of alphabets that we call Li-A( $N$ ) and Li-B( $N$ ), where  $N$  is the number of classes. A class in Li-B( $i$ ) is guaranteed to be a subset of a class in Li-B( $j$ ) if  $i \geq j$ ; in Li-A alphabets this is not the case: a class may split as  $N$  decreases, a phenomenon the authors call interlacing. For example, Li-A(5) has a class ILMV, while Li-A(4) has a class ILV; note that M has split out of the class despite the reduced number of classes. Solis and Rackovsky (8) minimized the information loss induced in the amino acid sequence and backbone conformation of known structures, again using a Monte Carlo search; we call their alphabets Solis-D( $N$ ) and Solis-G( $N$ ) [D = DSSP, G = GMBR; for definitions see (8)]. Murphy *et al.* (9) used a heuristic greedy algorithm based on correlation coefficients derived from BLOSUM50, giving alphabets we call Murphy( $N$ ). In related work, Taylor (10) presented an early analysis of amino acid groups in relation to alignments, and Wu and Brutlag (11) identified conserved amino acid groups in multiple sequence alignment databases; however, these groups are not required to be disjoint. In this work, the central observation is that

\*Email: bob@drive5.com

```

SRC2_XENLA      yvErmnyvHRDLrAaNiLvGENlvcKvaDFGLARl
STK_HYDAT      flEaggyiHRDLaArNiLvGENyicKvaDFGLARl
SRC1_DROME     ylEskqliHRDLaArNvLiGENnvaKicDFGLARv
SRK3_SPOLA     ylEahsyiHRDLaArNiLvGEgnvcKvaDFGLARv
FRK_HUMAN      ylEsrnyiHRDLaArNvLvGEhniyKvaDFGLARv
                *      **** * * * **      *      *****

SRC2_XENLA      FIDkidfIHKDIaAaDIIIGDdiIcKIaDFGIaKI
STK_HYDAT      FIDakgfIHKDIaAaDIIIGDdfIcKIaDFGIaKI
SRC1_DROME     FIDakkiIHKDIaAaDIIIGDddIaKIcDFGIaKI
SRK3_SPOLA     FIDahafIHKDIaAaDIIIGDgdIcKIaDFGIaKI
FRK_HUMAN      FIDakdfIHKDIaAaDIIIGDhdIfKIaDFGIaKI
                ***      ***** * ***** * ** *****

```

**Figure 1.** Here we show a core block from the CLUSTAL\_W alignment of selected sequences containing SH2 domains. The upper alignment uses A, the standard amino acid alphabet. The lower version is the same alignment presented in the SE-V(10) alphabet (Table 1) in which members of a class are represented by its first letter in alphabetical order (e.g. I, L, M and V are shown as I). Columns that are perfectly conserved in the given alphabet are indicated with upper case letters and an asterisk (\*) below. The number of conserved columns increases from 17 in A to 25 in SE-V(10). The number of fully conserved  $k$ -mers increases from 6 to 13 for  $k = 3$  and from 4 to 9 for  $k = 4$ .

identity is better conserved in a compressed alphabet (Fig. 1). Pairs of related sequences always have higher or equal identity, and hence tend to have more common  $k$ -mers, in an alphabet with  $N < 20$  than in A. If the alphabet is chosen such that there are high probabilities of intra-class substitution and low probabilities of inter-class substitution, then we might expect that the extent of identity in related regions could be usefully extended while limiting the increase in contiguous identities due to chance.

## METHODS

### Optimal alphabets from information theory

Substitution matrices can be viewed from an information theory perspective (12). The similarity score  $s_{ab}$  of amino acids  $a, b$  is understood to be derived from the joint probability  $p(a, b)$  of  $a$  and  $b$  being aligned to each other and the background probability  $p(a)$ , as follows:

$$s_{ab} = \log [p(a, b)/p(a) p(b)] \quad 1$$

Gavin Crooks (personal communication) has suggested that the evolutionary process implied by the substitution matrix should minimize the average information loss in a sequence, with the goal of making related sequences maximally similar in the resulting alphabet. This is achieved by maximizing the entropy of the compressed substitution matrix, defined as follows:

$$E(C) = \sum_{i \in C} \sum_{j \in C} p(i, j) \log [p(i, j)/p(i) p(j)] \quad 2$$

Consider a large database of pairwise alignments in which a pair of aligned residues is represented by a single symbol, with probability  $p(i, j)$ . The average information per symbol is then the entropy defined above, and Crooks' criterion for an optimal alphabet is understood as maximizing the information content of the database as expressed in that alphabet.

### Greedy algorithm for optimal alphabet search

Given a maximization function, there are too many possible compressed alphabets to be exhaustively tested on a conventional computer. We used a greedy algorithm that seeks to maximize the entropy (equation 2). Note that when  $N = |A| - 1$ , there are  $N(N - 1)/2$  compressed alphabets formed by making a class from each pair of letters. Starting with  $N = |A| - 1$ , we test all alphabets and keep the  $m$  best, where  $m$  is a parameter of the algorithm. This procedure is repeated on all retained alphabets with  $N$  reduced by 1, iterating until  $N = 2$ . If  $m$  is sufficiently large, the algorithm is exact but computationally intractable for interesting values of  $N$ . We used two substitution matrices for which the underlying probabilities are readily available: BLOSUM62 and the VTML240 matrix (13). (We found integer rounded log-score matrices to have inadequate precision.) We found convergence at  $m = 2$  (BLOSUM62) and  $m = 4$  (VTML240), meaning that identical alphabets were found using all greater values of  $m$  that we tried (up to 4906). This rapid and stable convergence suggests that the algorithm found global maxima. The alphabets produced by this method are denoted using SE-B( $N$ ) for BLOSUM62 and SE-V( $N$ ) for VTML240.

### $k$ -mer counting

A  $k$ -mer is a contiguous subsequence of length  $k$ , also known as a word or  $k$ -tuple. Related sequences tend to have more  $k$ -mers in common than expected by chance, provided that  $k$  is not too large and the divergence is not too great. Many sequence comparison methods based on  $k$ -mer counting have been proposed in the literature (reviewed in 14). The statistics of  $k$ -mers in sequences related by Markov processes were studied in detail by Reinert *et al.* (15). The number of  $k$ -mers shared by two sequences can be determined using a simple  $O(L)$  algorithm that assigns consecutive integer values to each possible  $k$ -mer and uses this integer as an index in a table. This technique was used by Jones *et al.* (16), who counted 3mers in A to identify closely related sequences.

### $k$ -mers in compressed alphabets

The MAFFT family of algorithms (17) constructs an initial guide tree for multiple alignment using a fast clustering method. The distance measure is obtained by counting 6mers in a compressed alphabet we call Dayhoff (6), which contains six amino acid classes identified by Dayhoff *et al.* (18). The authors' motivation for using a compressed alphabet and the choices of  $k$  and  $N$  are not discussed, but the idea is intuitively appealing. As sequences diverge, the number of common  $k$ -mers in A will on average be reduced, ultimately reaching a limit comparable to the number expected in unrelated sequences. If a compressed alphabet is used, we might expect this limit to be reached at a greater evolutionary distance, and perhaps suitable choices of  $N$  and  $k$  might give better estimates in different identity ranges.

### Estimating evolutionary distance

As a measure of the distance between two sequences we define the fractional common  $k$ -mer count  $F$  of two sequences 1 and 2, as follows:

$$F = \sum_{\tau} \min [n_1(\tau), n_2(\tau)] / [\min (L_1, L_2) - k + 1] \quad 3$$

Here  $\tau$  is a  $k$ -mer,  $n_1(\tau)$  and  $n_2(\tau)$  are the number of times  $\tau$  occurs in 1 and 2, respectively, and  $L_1, L_2$  are the sequence lengths. With appropriate choices of alphabet and  $k$ , this is equivalent to the measures used in (16) and by MAFFT. It may be motivated by a simple model that assumes homologous  $k$ -mers are always identical. Then  $\min [n_1(\tau), n_2(\tau)]$  is an upper bound on the number of instances of  $\tau$  that are homologous, and the denominator is the number of  $k$ -mers in the shorter sequence, i.e. an upper bound on the total number of potentially homologous  $k$ -mers. Common  $k$ -mers due to chance add positive noise to  $F$ ; mutations that cause a  $k$ -mer to change reduce  $F$ , which therefore tends to decrease with increasing evolutionary distance. Improvements might be achieved by weighting the average according to a feature expected to correlate with the probability that a common  $k$ -mer is homologous, such as its background probability or the difference in its positions in the two sequences compared with the difference seen in other common  $k$ -mers, but we do not consider these ideas further here.

### ***k*-mer extension**

All identical subsequences of length  $\geq k$  between two sequences can be identified using the following procedure: assign a unique integer  $0 \dots (N^k - 1)$  to a  $k$ -mer by interpreting letters as digits with radix  $N$ . Use this integer as an index in a table, with one entry for each possible  $k$ -mer. An entry in this table contains a list of positions where the corresponding  $k$ -mer is found. The table is first built for one sequence.  $k$ -mers in the second sequence are then enumerated in position order until a common  $k$ -mer is found. The position(s) of that  $k$ -mer in the second sequence are determined using the table, and each match extended one letter at a time until the sequences differ. Enumeration of  $k$ -mers in the second sequence then resumes from the end of the longest match. We call this method  $k$ -mer extension. A compressed alphabet may be used with the expectation that longer matches will be found. The matches found by this method are candidate local alignments. If desired, an expectation value can be computed for a matched region. Diagonals may be further extended by allowing high-scoring substitutions rather than matching letters, or by introducing full dynamic programming (DP) optimization on the remaining regions.  $k$ -mer extension is related to techniques employed by several well-known algorithms [e.g. FASTA (19) and BLAST] to find and extend high-scoring diagonals. In MAFFT, the fast Fourier transform (FFT), which has  $O(L \log L)$  time complexity, is used. We propose  $k$ -mer extension as a significantly faster alternative to FFT. If the list contained in a table entry is allowed to be of arbitrary length, the algorithm is properly  $O(L^2)$ , although we find that the effective complexity for typical sequences is close to  $O(L)$  as multiple instances of a given  $k$ -mer are relatively rare in a suitable alphabet. This can be addressed by accepting  $O(L^2)$  as a worst case, or by setting a maximum length for the list of positions where a given  $k$ -mer is found, which restores  $O(L)$  at the cost of allowing false negatives. In the latter case, a tie-breaker is applied when the maximum is exceeded. When used to optimize global alignment, it is useful to keep the position(s) closest to the center of the sequence, as the saved DP matrix area is increased by a central diagonal versus one near a terminal. Also, terminal regions are less likely to contain diagonals. Application of the method to profile

alignment is straightforward. If a column in a multiple alignment is fully conserved in the chosen alphabet, it is represented by the appropriate letter. Otherwise it is represented by a new symbol that fails to match all letters and itself.

## **RESULTS**

We used a set of 1484 pairs of protein structures selected by Sadreyev and Grishin (20) (data kindly provided by Ruslan Sadreyev). These pairs were chosen to be representative of alignable structures in the FSSP database (21).

### **The *k*-mer distance**

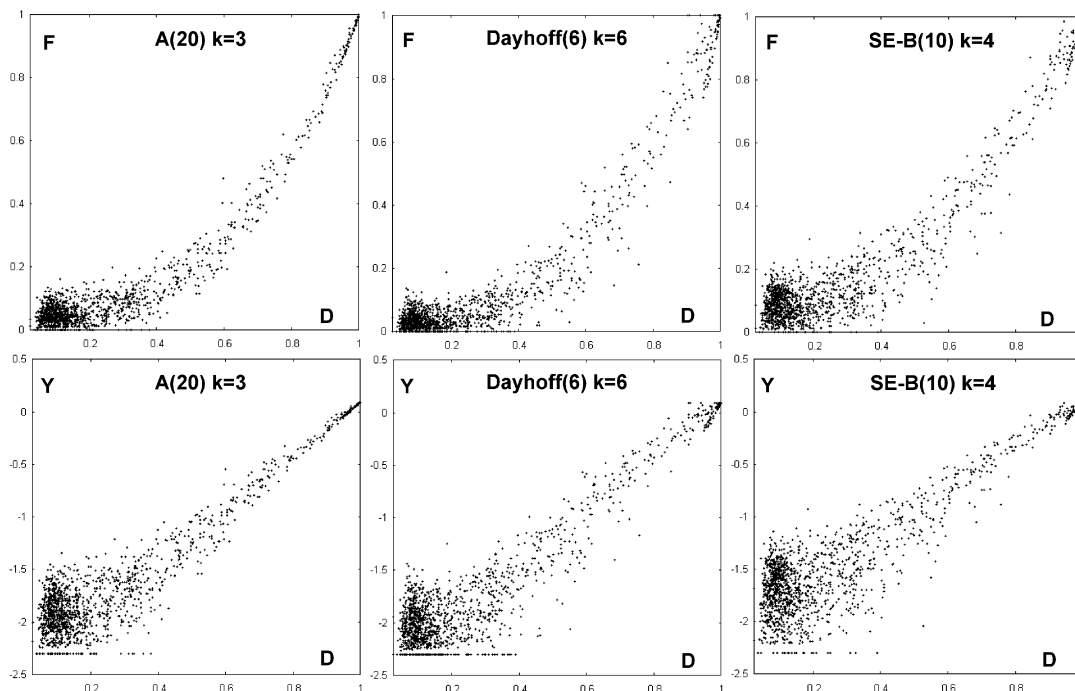
In order to evaluate the performance of  $k$ -mer counting we need an objective measure of distance. We are interested in the application of  $k$ -mer counting to fast clustering in global multiple sequence alignment algorithms, and therefore choose to use the fractional identity  $D$  computed from the CLUSTAL\_W alignment of two full-chain sequences for this purpose.  $D$  is used by CLUSTAL\_W to construct its own guide tree. However, it should be noted that identity is an approximate distance measure that is increasingly unreliable at lower identities (22). Possibly a distance measure derived by  $k$ -mer counting could be more accurate than identity for sufficiently diverged sequences, but this is not readily testable due to the lack of reference data for true distances. Starting from  $F$ , as defined in equation 3, we find empirically that a logarithmic transformation yields an approximately linear relationship with  $D$  for a wide range of alphabets and values of  $k$ . The transformed measure  $Y$ , which we call the  $k$ -mer distance, is defined as follows:

$$Y = \log(0.1 + F) \quad 4$$

We use the correlation coefficient  $r$  between the  $k$ -mer distance  $Y$  and identity  $D$  as a quality score: we interpret a greater  $r$  as indicating a better measure. Figure 2 shows scatter plots of  $F$  and  $Y$  for three choices of parameters (alphabet and  $k$ ). We computed  $r$  for values of  $k$  from 2 to 8 and the following alphabets: Solis-D(2–14), Solis-G(2–14), Li-A(2–19), Li-B(2–19), SE-B(2–19), SE-V(2–19), Murphy(8,10,15) and Dayhoff(6), giving a total of 86 unique alphabets after duplicates are eliminated. Table 1 shows some representative alphabets. Table 2 shows  $r$  for these alphabets and selected values of  $k$  for the complete set of sequence pairs and also for a subset of low identity ( $D = 0.25$ – $0.5$ ). We made no attempt to assess the statistical significance of the rankings implied by these results as the reference standard  $D$  is itself approximate, and errors in predicting  $D$  are larger than the typical standard deviation. Surprisingly, we find little evidence that compressed alphabets give any improvement over the full alphabet A in any range of identities. This is illustrated by the low identity results for  $D = 0.25$ – $0.5$  (Table 2), where the  $k$ -mer distances computed with A,  $k = 4$  produce the best correlation with  $D$  of all the tested combinations (complete results not shown).

### **Phylogenetic tree construction**

The  $k$ -mer distance can be used to rapidly construct phylogenetic trees. Given  $N$  unaligned sequences of length  $L$ , the  $k$ -mer distance  $Y$  is calculated for each pair, giving an



**Figure 2.** Scatterplots show the correlation between  $D$ , the fractional identity in the full alphabet computed from CLUSTAL\_W alignments, and the fractional common  $k$ -mer count ( $F$ , equation 3) or the  $k$ -mer distance ( $Y$ , equation 4). We show three selected cases: the full alphabet A,  $k = 3$  [the parameters used in (16)]; Dayhoff(6),  $k = 6$  (used by MAFFT); and an intermediate case, SE-B(10),  $k = 4$ . Note that the relationship between  $D$  and  $Y$  is approximately linear.

**Table 1.** Examples of compressed alphabets produced by different methods

Alpha( $N$ )	Classes
SE-B(14)	A, C, D, EQ, FY, G, H, IV, KR, LM, N, P, ST, W
SE-B(10)	AST, C, DN, EQ, FY, G, HW, ILMV, KR, P
SE-V(10)	AST, C, DEN, FY, G, H, ILMV, KQR, P, W
Li-A(10)	AC, DE, FWY, G, HN, IV, KQR, LM, P, ST
Li-B(10)	AST, C, DEQ, FWY, G, HN, IV, KR, LM, P
Solis-D(10)	AM, C, DNS, EKQR, F, GP, HT, IV, LY, W
Solis-G(10)	AEFIKLMQRVW, C, D, G, H, N, P, S, T, Y
Murphy(10)	A, C, DENQ, FWY, G, H, ILMV, KR, P, ST
SE-B(8)	AST, C, DHN, EKQR, FWY, G, ILMV, P
SE-B(6)	AST, CP, DEHKNQR, FWY, G, ILMV
Dayhoff(6)	AGPST, C, DENQ, FWY, HKR, ILMV

Alphabet names are defined in the main text.

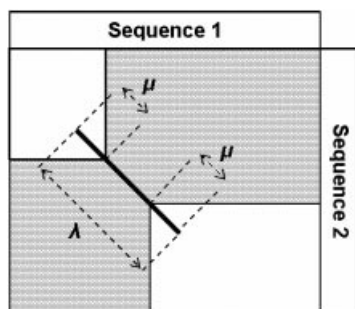
estimate  $D$  of the fractional identity. This is converted to an estimate  $d$  of an additive evolutionary distance, for example by using the Kimura correction:  $d = \ln [D - (1 - D^2)/5]$  (22). The resulting distance matrix is then used to construct a tree through a clustering algorithm such as UPGMA (2) or neighbor-joining (3). This gives a distance matrix in  $O(LN^2)$  time, versus  $O(L^2N^2)$  for typical measures obtained from pairwise alignments. As a practical illustration, we chose a set of 2000 sequences (average length 282) obtained by a search for homologs of dienoyl-coenzyme A isomerase. Starting from unaligned sequences, CLUSTAL\_W's distance matrix computation on this set required 6.2 h. Our preliminary  $k$ -mer distance implementation was able to compute a matrix in 6.1 s, faster by a factor of 3700. (All times quoted in this work were measured using a 2.5 GHz Pentium 4 processor.) The quality

**Table 2.** Measured correlation coefficients ( $r$ ) for  $k = 3$  to 7 between the  $k$ -mer distance ( $Y$ ) and fractional identity ( $D$ ) for the full alphabet A and the alphabets described in Table 1

Alpha( $N$ )	$r$ (all pairs)							$r$ ( $D = 0.25-0.5$ )		
	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$	$k = 3$	$k = 4$	$k = 5$	$k = 6$	$k = 7$
A(20)	0.943	0.962	0.944	0.926	0.910	0.575	0.685	0.631	0.584	0.520
SE-B(14)	0.881	0.964	0.961	0.948	0.933	0.422	0.634	0.649	0.611	0.547
SE-B(10)	0.771	0.907	0.959	0.961	0.951	0.248	0.455	0.585	0.589	0.563
SE-V(10)	0.746	0.880	0.951	0.962	0.956	0.232	0.409	0.570	0.602	0.598
Li-A(10)	0.842	0.956	0.964	0.953	0.939	0.350	0.587	0.652	0.634	0.593
Li-B(10)	0.793	0.933	0.962	0.956	0.943	0.281	0.513	0.605	0.592	0.553
Solis-D(10)	0.776	0.913	0.952	0.961	0.933	0.261	0.505	0.625	0.654	0.612
Solis-G(10)	0.692	0.730	0.763	0.812	0.858	0.176	0.205	0.228	0.284	0.323
Murphy(10)	0.770	0.910	0.960	0.962	0.952	0.240	0.441	0.579	0.607	0.592
SE-B(8)	0.751	0.869	0.944	0.962	0.956	0.222	0.396	0.558	0.602	0.588
SE-B(6)	0.711	0.797	0.885	0.939	0.954	0.215	0.312	0.458	0.555	0.580
Dayhoff(6)	0.638	0.737	0.861	0.937	0.955	0.165	0.254	0.407	0.540	0.589

Correlations with  $k = 2$  are lower, the highest for all pairs being  $r = 0.739$  using A. The  $k = 1$  case reduces to a simple composition bias.

of the resulting trees is hard to assess for the usual reason that reference data are generally not available, but given the good correlation between  $k$ -mer distance and identity we expect that in many cases they will be of comparable quality or only slightly degraded compared with those derived from aligned identity. We further expect that the  $k$ -mer distance estimate of identity will tend to have systematic errors within a given protein family, and will therefore exhibit reduced scatter and improved correlation compared with data from unrelated families (as in Fig. 2). Note that edge lengths are unimportant in a tree used to guide multiple alignment, which requires only that the branching order is approximately correct.



**Figure 3.** The geometry of a diagonal (bold line) in a dynamic programming matrix. The length of the diagonal is  $\lambda$ , the margin is  $\mu$ . If the diagonal can be identified in a pre-processing step such as  $k$ -mer extension or FFT, then the shaded region can be excluded, reducing the time needed to compute a global alignment by dynamic programming.

**Table 3.** Selected results for  $k$ -mer extension with  $\lambda \geq 8$ ,  $\mu \geq 3$

Alpha( $N$ )	$k$	Coverage	$\psi$
Li-A(9)	6	30 868	0.995
SE-V(12)	5	30 713	0.996
Li-A(9)	5	30 526	0.995
SE-V(11)	4	30 519	0.995
Li-A(10)	6	30 478	0.995
SE-V(12)	5	30 474	0.995
Li-B(9)	4	30 472	0.995
Li-B(10)	4	30 460	0.995
A(20)	4	26 759	0.999
A(20)	3	26 494	0.999

Coverage is the number of correct positions, determined by reference to the 465 907 positions in the CLUSTAL\_W alignments. The fraction of positions that are correct according to CLUSTAL\_W is  $\psi$ . Included are the best eight combinations of alphabet and  $k$  satisfying  $\psi \geq 0.995$  and  $N^k$  (lookup table size)  $\leq 10^6$ ; also, A with  $k = 3$  and  $k = 4$  for comparison. Rows are sorted by coverage.

### Optimizing global alignment time

If diagonals are known, they can be used to reduce alignment time. Our focus is again on global alignment, where a diagonal reduces the DP matrix to two submatrices (Fig. 3). Note that even a short match can give significant time savings; for example, the matrix area is at least halved by a diagonal of length  $\geq 1$  located in the midpoint of both sequences. We evaluated the ability of  $k$ -mer extension to correctly identify diagonals in the CLUSTAL\_W alignments of the 1848 Sadreyev–Grishin pairs. (The FSSP structural alignments were not used as they are local rather than global and also ignore sequence homology, so regions of high similarity are sometimes shifted.) We denote the length of a  $k$ -mer extension match by  $\lambda$  and the margin, defined as the distance of a given letter from the closest end of the match, by  $\mu$  (see Fig. 3). We find, as would be expected, that the fraction  $\psi$  of correct match positions increases with  $\lambda$  and with  $\mu$ , resulting in a trade-off between coverage (number of correct match positions) and error (false-positive positions). For example, with  $\lambda \geq 8$  and  $\mu \geq 3$ , we find that many choices of alphabet and  $k$  give  $\psi > 0.99$ . It is then reasonable to include the entire match (minus margins  $< \mu$ ) rather than just the central position in the

diagonal, as MAFFT does. Some typical results are shown in Table 3. As before, we do not attempt to assess the statistical significance of differences in performance.

These results strongly suggest that use of a suitable compressed alphabet can improve performance, with the better parameters giving  $\sim 15\%$  more coverage over the full alphabet A. To compare performance against the FFT, we modified MAFFT to record the reduction in DP matrix area and execution time of the routines that find diagonals. With the Sadreyev–Grishin pairs, we found that MAFFT reduced the total DP space from 117 136 137 to 92 716 633 matrix elements, a reduction of 21%. Total execution time for diagonal finding was 2.4 s, mostly consumed by the FFT routines. Using  $k$ -mer extension with SE-V(11),  $k = 4$ , the DP space was reduced to 97 099 841 elements, saving 17%. Our preliminary implementation allowed only one position per  $k$ -mer; we believe the coverage can be improved with minimal cost in time by allowing multiple positions. Total execution time for finding diagonals was 0.08 s. Time spent performing DP in MAFFT was 7.1 s, compared with 9.0 s. with FFT disabled. Time for diagonal finding and DP combined was therefore  $7.1 + 2.4 = 9.5$  s, demonstrating that on this test data the net result of using FFT is an increase in execution time of  $\sim 5\%$ . If FFT were replaced by  $k$ -mer extension, the total time would be reduced to  $7.1 + 0.08 = 7.2$  s, saving 20%. Of the 0.5%  $k$ -mer extension positions that were incorrect according to CLUSTAL\_W, none were in the regions aligned according to FSSP, and hence were found exclusively in regions where there may be no meaningful sequence alignment (due to lack of homology and/or divergent structure).

### DISCUSSION

We present a method for discovering local similarities between two sequences by identifying a common  $k$ -mer and extending the match. This is achieved by a fast, simple algorithm that has  $O(L)$  time complexity. We show that use of a compressed amino acid alphabet can increase the coverage of the method with a negligible increase in errors compared with full dynamic programming. On a test set of 1848 sequence pairs selected by Sadreyev and Grishin from the FSSP database, we find that this method achieves comparable coverage to the FFT method used by MAFFT and is more than an order of magnitude faster. On this test, we find that using FFT fails to save time, giving a net increase of  $\sim 5\%$ . Replacing FFT with  $k$ -mer extension would give a reduction of 20% in time. Greater reduction would be expected when aligning more closely related sequences. We also investigate the use of  $k$ -mer counting as a fast estimate of evolutionary distance. We show that  $k$ -mer distances correlate well with the fractional identity computed from a global alignment. However, contrary to our expectations, we fail to find evidence that use of a compressed alphabet can improve the accuracy of the estimate. We suggest the use of  $k$ -mer distance as the basis for rapid construction of phylogenetic trees. This reduces the time complexity of distance matrix construction from  $O(L^2N^2)$  to  $O(LN^2)$ , which we show can lead to a time reduction in this component of practical applications of more than three orders of magnitude.

## ACKNOWLEDGEMENTS

The author is grateful to Doug Brutlag, Gavin Crooks, Richard Durbin and Tobias Müller for helpful discussions. Ruslan Sadreyev generously provided the FSSP test data.

## REFERENCES

- Altschul,S.F., Gish,W., Miller,W., Myers,E.E. and Lipman,D.J. (1990) Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410.
- Sneath,P.H.A. and Sokal,R.R. (1973) *Numerical Taxonomy*. Freeman, San Francisco, CA.
- Saitou,N. and Nei,M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.*, **4**, 406–425.
- 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.
- Durbin,R., Eddy,S., Krogh,A. and Mitchison,G. (1998) *Biological Sequence Analysis*. Cambridge University Press, Cambridge, UK.
- Li,T., Fan,K., Wang,J. and Wang,W. (2003) Reduction of protein sequence complexity by residue grouping. *Protein Eng.*, **16**, 323–330.
- Henikoff,S. and Henikoff,J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl Acad. Sci. USA*, **89**, 10915–10919.
- Solis,A.D. and Rackovsky,S. (2000) Optimized representations and maximal information in proteins. *Proteins*, **38**, 149–164.
- Murphy,L.R., Wallqvist,A. and Levy,R.M. (2000) Simplified amino acid alphabets for protein fold recognition and implications for folding. *Protein Eng.*, **13**, 149–152.
- Taylor,W.R. (1982) The classification of amino acid conservation. *J. Theor. Biol.*, **119**, 205–208.
- Wu,T.D. and Brutlag,D.L. (1996) Discovering empirically conserved amino acid substitution groups in databases of protein families. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, **4**, 230–240.
- Altschul,S.F. (1991) Amino acid substitution matrices from an information theoretic perspective. *J. Mol. Biol.*, **219**, 555–565.
- Müller,T., Spang,R. and Vingron,M. (2002) Estimating amino acid substitution models: a comparison of Dayhoff's estimator, the resolvent approach and a maximum likelihood method. *Mol. Biol. Evol.*, **19**, 8–13.
- Vinga,S. and Almeida,J. (2003) Alignment-free sequence comparison—a review. *Bioinformatics*, **19**, 513–523.
- Reinert,G., Schbath,S. and Waterman,M.S. (2000) Probabilistic and statistical properties of words: an overview. *J. Comp. Biol.*, **7**, 1–46.
- Jones,D.T., Taylor,W.R. and Thornton,J.M. (1992) The rapid generation of mutation data matrices from protein sequences. *CABIOS*, **8**, 275–282.
- Katoh,K., Misawa,K., Kuma,K. and Miyata,T. (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.
- Dayhoff,M.O., Schwartz,R.M. and Orcutt,B.C. (1978) A model of evolutionary change in proteins. In Dayhoff,M.O. and Ech,R.V. (eds), *Atlas of Protein Sequence and Structure*. National Biomedical Research Foundation, MD, pp. 345–352.
- Pearson,W.R. and Lipman,D.J. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **4**, 2444–2448.
- Sadreyev,R. and Grishin,N. (2003) COMPASS: a tool for comparison of multiple protein alignments with assessment of statistical significance. *J. Mol. Biol.*, **326**, 317–336.
- Holm,L. and Sander,C. (1998) Touring protein fold space with Dali/FSSP. *Nucleic Acids Res.*, **26**, 316–319.
- Kimura,M. (1983) *The Neutral Theory of Molecular Evolution*. Cambridge University Press, Cambridge, UK.