Multiple sequence alignment with the divide-and-conquer method

Jens Stoye *

Research Center for Interdisciplinary Studies on Structure Formation (FSPM), University of Bielefeld, Postfach 10 01 31, D-33501 Bielefeld, Germany

Abstract

An improved algorithm for the simultaneous alignment of multiple protein and nucleic acid sequences, the Divide-and-Conquer Alignment procedure (DCA), is presented. The basic method described in Tönges et al. (1996) (Tönges, U., Perrey, S.W., Stoye, J., Dress, A.W.M., 1996. A general method for fast multiple sequence alignment. Gene, 172, GC33–GC41) is generalized to align any number of sequences and to work with arbitrary (e.g. affine linear) gap penalty functions. Also, the practical efficiency of the method is improved so that families of more than 10 sequences can now be aligned simultaneously within a few seconds or minutes. After a brief description of the general method, we assess the time and memory requirements of our implementation of DCA. We present several examples showing that the program is able to deal with real-world alignment problems.

Keywords: Divide-and-Conquer Alignment; Multiple Sequence Alignment

1. Introduction

Basically, all methods for sequence alignment are based, one way or another, on the dynamic programming algorithm of Needleman and Wunsch (1970). Yet, while it is, in theory, very simple and elegant, the generalization of the pairwise method to simultaneous multiple sequence alignment is computationally demanding and becomes—despite much work on improving this situation—impracticable for about six and more sequences of relevant length. Moreover, with the NP completeness of multiple sequence alignment (Wang and Jiang, 1994), any attempt at developing a fast algorithm for the computation of optimal alignments of many sequences is expected to fail. Consequently, there is a great need for heuristic algorithms producing near-optimal alignments, and an abundance of procedures have been developed. For reviews and comparisons, see Argos et al. (1991), Chan et al. (1992), Pevzner (1992) and McClure et al. (1994). Existing approaches generally fall into one of the following two classes. Progressive alignment methods iteratively align pairs of sequences or already-aligned subfamilies (so-called profiles or average sequences) guided by the branching order of a pre-given (mostly unrooted) tree whose leaves represent the sequences. Algorithms that fall into the second class, fragment-based methods, follow the strategy of assembling pairwise or multiple local alignments. After a consistency check, the local alignments define fixed regions or anchors of the intended global alignment. The remaining subsequences between the anchors are then aligned optimally.

The Divide-and-Conquer Alignment method that we describe and assess in this paper can (in some sense) be seen as flowing from the same concept as the fragment-based methods but being more general than previous procedures. Systematically, anchor points are fixed in all of the sequences, whether there are obvious local similarities or not. Hence, a considerable increase in speed compared to optimal multiple alignment by dynamic programming can be guaranteed.

With restricted functionality, the method has been previously presented in Tönges et al. (1996) and Stoye et al. (1997a), and a thorough discussion of the algorithm can be found in Stoye (1997). In this paper, we show how the method is generalized to more than three sequences and how general gap costs are handled, making the resulting computer program DCA now

* Present address: University of California at Davis, Department of Computer Science, Davis, CA 95616, USA. Tel: +1 530 754 8742; Fax: +1 530 752 4767; e-mail: stoye@cs.ucdavis.edu

Abbreviations: DCA, Divide-and-Conquer Alignment; MSA, Multiple Sequence Alignment; SP, Sum of Pairs.

applicable to real-world alignment problems. We also present alignments computed with DCA of several frequently used benchmark problems from the literature.

DCA is freely available on-line from the address http://bibiserv.TechFak.Uni-Bielefeld.DE/dca/.

2. Materials and methods

In the first part of this section, we briefly state the multiple sequence alignment problem whose solution we aim to approximate. Then, the DCA procedure is presented in its generalized form, followed by a closer look at the 'heart' of the procedure, how cut positions are computed. For details regarding the implementation, we refer to other publications.

2.1. Simultaneous multiple sequence alignment

For various reasons, it is our objective to align—in contrast to progressive alignment methods—the sequences simultaneously, i.e. we do not presuppose a phylogenetic tree of the sequences as the basis of our alignment. This can have—at least in principle—advantages, in particular when the alignments are used to reconstruct phylogenetic relationships of the involved sequences: it has often been noticed that the order of progressive sequence alignment can bias the alignment towards exactly that phylogeny which was used as a basis for the alignment (Lake, 1991; Thorne and Kishino, 1992; Hein, 1994). The simultaneous alignment approach avoids such circularities.

It has also been argued that alignment errors in early stages of progressive methods cannot be corrected when more information about the true situation becomes available. Feng and Doolittle (1987) coined the term 'once a gap, always a gap'. Such problems are also avoided by our approach since the full information from all of the sequences is taken into account already in the first alignment step.

Among objective functions for simultaneous multiple sequence alignment, in the last decade, the so-called sum-of-pairs (SP) score, defined as the sum of the scores of all induced pairwise alignments, has received a large amount of attention (Carrillo and Lipman, 1988; Altschul and Lipman, 1989; Gusfield, 1993; Gotoh, 1996). Sometimes, the pairwise costs are additionally weighted according to sequence-dependent (non-negative) weight factors (Altschul and Erickson, 1986; Gotoh, 1996; Ben-Dor et al., 1997) to avoid overweighting redundant information that can arise, e.g. from some identical or very similar sequences in the data set.

Formally, we will consider the multiple sequence alignment problem in the following form: Assume a family of k sequences $s_1,\ldots,s_k$. Let $\mathrm{A}(s_1,\ldots,s_k)$ denote the set of all alignments of $s_1,\ldots,s_k$. Then, given a pairwise alignment weight function, w_2, find an alignment $A \in \mathrm{A}(s_1,\ldots,s_k)$ with minimal weight:

$$w(A) := \sum_{1 \le p < q \le k} \alpha_{p,q} \cdot w_2(s_p^*, s_q^*),$$

where $\alpha_{p,q}$ are weight factors as discussed above, and s_p^* and s_q^* are the aligned pth and qth sequence. In our general description of DCA, we consider pairwise functions w_2 with arbitrary length-dependent gap penalty functions $g(l)$. The current implementation is restricted to affine gap costs of the form $g(l) := \alpha + \beta l$ for a gap of length l, which are generally considered appropriate for biological sequences. We assume the cost α for opening a gap and β for each symbol in the gap to be non-negative numbers.

2.2. Divide-and-Conquer Alignment Algorithm

The general idea of DCA is the following: Suppose, as above, that we are given a family of sequences $s_1,\ldots,s_k$. First, one of the sequences, say s_1 (in our current implementation, we always select the longest one), is cut at position c_1 near its midpoint. Then, depending on this choice, the remaining sequences $s_2,\ldots,s_k$ are cut at suitably fitting positions, say, sequence s_2 is cut at position c_2, s_3 is cut at position c_3, and so on. In this way, two new families of shorter sequences are obtained, one family consisting of the prefixes $s_1(\le c_1),\ldots,s_k(\le c_k)$ and one family consisting of the suffixes $s_1(>c_1),\ldots,s_k(>c_k)$. Here, $s(\le c)$ denotes the (prefix) subsequence of s with indices running from 1 to c, and $s(>c)$ denotes the (suffix) subsequence of s with indices running from $c+1$ to $|s|$ where $|s|$ denotes the length of sequence s. If these two new families of sequences could be aligned optimally, then by simple concatenation of the resulting alignments, an alignment of the original sequences could be obtained that is expected to be quite good if the cut sites are chosen carefully. However, if it still takes too much time to align these two new families optimally, the procedure can be applied in a recursive manner both to the prefix and to the suffix family. In this way, the original multiple alignment problem is divided into an increasing number of alignment problems involving shorter and shorter sequences, until the (sub)sequences are sufficiently short (e.g. shorter than a threshold, L, the so-called recursion stop size) so that they can be aligned optimally. Finally, the remaining short alignments of the subsequences are concatenated, yielding a solution of the original alignment problem.

By this recursive procedure, the problem of aligning k sequences of length at most n is reduced to the problem of aligning about n/L families of short (sub)sequences of maximal length L. For a schematic representation of the divide-and-conquer method for three sequences, see Fig. 1.

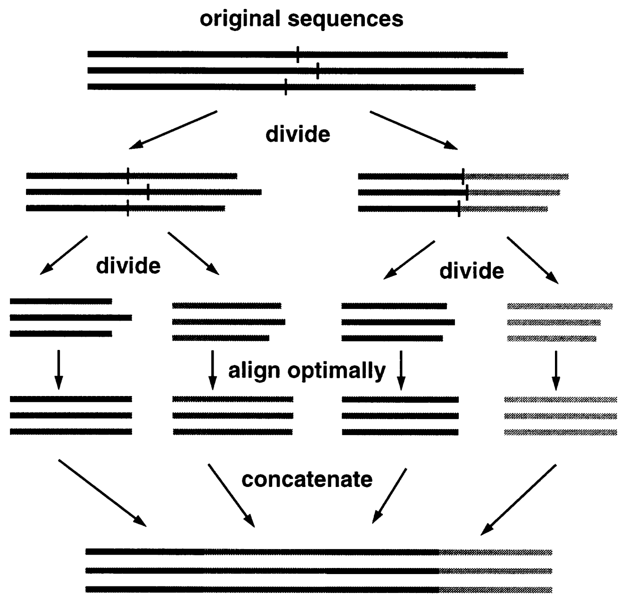

Fig. 1. Divide-and-conquer alignment algorithm.

Fig. 2 gives an impression of the reduction of search space achieved: Suppose each of the three sequences is represented by a set of parallel edges of the large box in Fig. 2a. Then, the size of the corresponding alignment problem is proportional to the volume of this box. By cutting the sequences, the large problem is reduced to several smaller alignment problems, represented by the 'chain' of boxes along the main diagonal of the large box (see Fig. 2b and c). The remaining search space is then the sum of the volumes of these small boxes.

For efficiency reasons, DCA uses the widely known program MSA (Lipman et al., 1989; Gupta et al., 1995) for aligning the families of remaining short subsequences. Therefore, the current implementation of DCA can be seen as a wrapper for MSA, although, in principle, any other multiple alignment program could be used here.

2.3. Computing cut positions

Of course, the main difficulty arising with DCA is how to find suitable cut positions such that the resulting total alignment is optimal or, at least, close to an optimal alignment of the original sequences.

First, notice that for any cut position c_1 of sequence s_1, there always exist positions $c_2,...,c_k$, such that there are alignments, A of the family of prefixes and B of the family of suffixes, so that the concatenation of A and B, short $A++B$, is an optimal alignment of the original sequences. However, due to the NP completeness of multiple sequence alignment, one cannot expect to find a way computing such optimal cut positions efficiently. Instead, we proposed a heuristic method for an efficient computation of good—though not necessarily optimal—cut positions (Tönges et al., 1996), formulated in the following for the weighted SP score with an arbitrary pairwise alignment cost function w_2.

First, consider a pair of sequences (s,t). For each possible choice of the cut positions (i,j), $0 \leq i \leq |s|$, $0 \leq j \leq |t|$, we define the pairwise additional cost with respect to the pairwise cost function w_2 by

$$C_{s,t}(i,j) := \min \{w_2(A ++ B) | A \in \mathcal{A}[s(\leq i), t(\leq j)], \\ B \in \mathcal{A}[s(>i), t(>j)]\} - w_2^{\mathrm{opt}}(s,t),$$

where $w_2^{\mathrm{opt}}(s,t)$ denotes the optimal (i.e. minimal) alignment cost of s and t. The matrix

$$C_{s,t} := [C_{s,t}(i,j)]_{0 \leq i \leq |s|, 0 \leq j \leq |t|}$$

is called the additional-cost matrix of s and t with respect to w_2.

Fig. 3 illustrates the definition: Let an optimal alignment path with cost $w_2^{opt}(s,t)$ be represented by the chain of light shaded boxes and a best alignment path passing through vertex (i,j) be represented by the dark shaded boxes. The additional cost is simply the 'length difference' of these two paths.

Note that in case of an additive alignment score function w_2, the above definition of the pairwise additional cost is equivalent to that given in Tönges et al. (1996):

$$\begin{aligned} C_{s,t}[i,j] &= \min \{w_2(A ++ B) | A \in \mathcal{A}[s(\leq i), t(\leq j)], \\ &\quad B \in \mathcal{A}[s(>i), t(>j)]\} - w_2^{\mathrm{opt}}(s,t) \\ &= \min \{w_2(A) + w_2(B) | A \in \mathcal{A}[s(\leq i), t(\leq j)], \\ &\quad B \in \mathcal{A}[s(>i), t(>j)]\} - w_2^{\mathrm{opt}}(s,t) \\ &= \min \{w_2(A) | A \in \mathcal{A}[s(\leq i), t(\leq j)], + \min \{w_2(B) | \\ &\quad B \in \mathcal{A}[s(>i), t(>j)]\} - w_2^{\mathrm{opt}}(s,t) \\ &= w_2^{\mathrm{opt}}[s(\leq i), t(\leq j)] + w_2^{\mathrm{opt}}[s(>i), t(>j)] - w_2^{\mathrm{opt}}(s,t). \end{aligned}$$

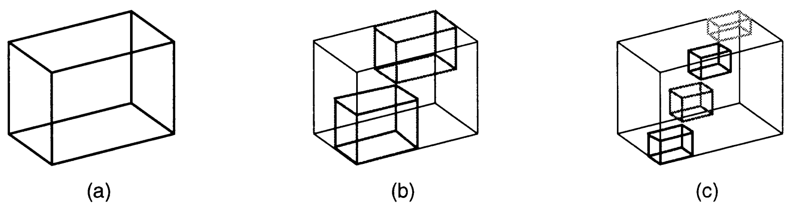

Fig. 2. Reduction of search space.

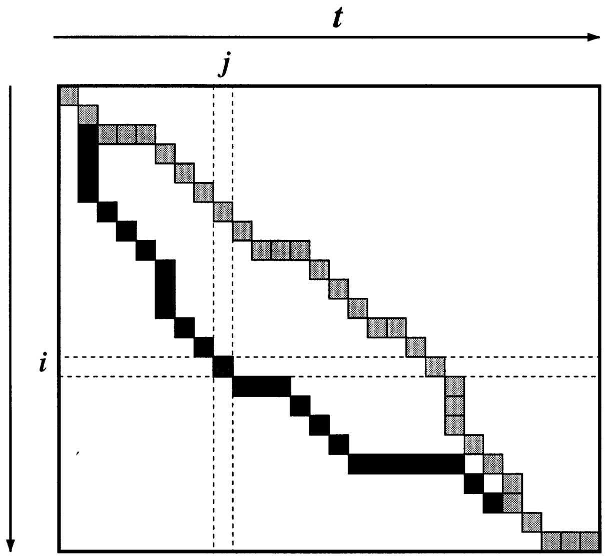

Fig. 3. Definition of $C_{s,t}$. Light shaded boxes denote an optimal alignment path, and dark shaded boxes denote a best alignment path through the vertex (i,j) (the black box).

For this case, it is shown in Tönges et al. (1996) that $C_{s,t}$ can be computed efficiently by a forward and backward pass over the alignment matrix, similar to approaches developed, for example, by Vingron and Argos (1990) in the context of dot plots and by Waterman (1983) to compute near-optimal alignments.

Yet, also for affine gap costs of the form $g(l)=\alpha+\beta l$, we can establish an algorithm that runs in time proportional to $|s|\cdot|t|$. Using two auxiliary matrices $H_{s,t}$ and $V_{s,t}$, Gotoh (1982) showed how to compute 'ordinary' alignments of two sequences s and t with affine gap costs in quadratic time:

$$V_{s,t}(i,j)=\min\,[D_{s,t}(i-1,j)+\alpha, V_{s,t}(i-1,j)]+\beta$$

$$H_{s,t}(i,j)=\min\,[D_{s,t}(i,j-1)+\alpha, H_{s,t}(i,j-1)]+\beta$$

$$D_{s,t}(i,j)=\min\begin{bmatrix} D_{s,t}(i-1,j-1)+d(s_i,t_j), \\ V_{s,t}(i,j), H_{s,t}(i,j) \end{bmatrix}$$

with initializations

$$D_{s,t}(0,0)=0,$$

$$V_{s,t}(0,0)=H_{s,t}(0,0)=+\infty,$$

$$D_{s,t}(i,0)=V_{s,t}(i,0)=g(i),$$

$$H_{s,t}(i,0)=+\infty,$$

$$D_{s,t}(0,j)=H_{s,t}(0,j)=g(j),$$

$$V_{s,t}(0,j)=+\infty$$

for all i, $1\leq i\leq|s|$ and j, $1\leq j\leq|t|$.

Using the three matrices V, H and D as well as the corresponding reverse matrices V^r, H^r and D^r, which are computed in a similar fashion (running the dynamic programming procedure in the reverse direction), the additional-cost matrix $C_{s,t}$ for affine gap penalties is computed in time proportional to $|s|\cdot|t|$:

$$C_{s,t}i,j=\min\begin{bmatrix} V_{s,t}(i,j)+V^r_{s,t}(i,j)-\alpha, \\ H_{s,t}(i,j)+H^r_{s,t}(i,j)-\alpha, \\ D_{s,t}(i,j)+D^r_{s,t}(i,j) \end{bmatrix}$$

for all (i,j), $0\leq i\leq|s|$, $0\leq j\leq|t|$. In the first two cases, the gap open penalty α is subtracted from the sum of the forward and reverse matrix entries because here upon concatenation, a gap at the right terminus of the left hand alignment merges with a gap at the left terminus of the right hand alignment resulting in a single gap crossing the cut position.

We now return to our original problem of computing suitable cut positions simultaneously for all of the sequences $s_2,\ldots,s_k$ given a cut position c_1 of sequence s_1. To this end, we compute—in analogy to the SP alignment score—the weighted sum over all pairwise additional-cost matrix entries. Our heuristic is that combinations $(c_2,\ldots,c_k)$ minimizing the value

$$C(c_1;c_2,\ldots,c_k):=\sum_{1\leq p<q\leq k}\alpha_{p,q}C_{s_p,s_q}(c_p,c_q)$$

yield good, if not optimal cut positions. Yet, finding the minimum of this value is itself a non-trivial problem, and several heuristics based on a method described in Stoye et al. (1997a) have been developed to speed up the procedure. Details can be found in Stoye (1997). For a general description of the current implementation of DCA, see Stoye et al. (1997b).

3. Results

In the first part, we discuss the general behavior of DCA depending on several independent parameters such as the number and the average length of the sequences. Since nature does not provide 'benchmark' problems suited for all considerable problem instances, we decided to perform these experiments on artificially created related random sequences[1]. We therefore developed a method to create sequence families simulating an evolutionary process by iterated mutation of a common ancestor sequence following the edges of a pre-given rooted mutation guide tree (Stoye et al., in press).

After the general discussion, several example alignments of protein sequence families from the literature are presented.

[1] Recently, a study consisting of a broad range of multiple sequence alignment problems has been published (Gotoh, 1996). However—although a valuable source for test cases of various kind—even these sequence families do not equally cover the whole sequence space and hence are not suitable to assess the general time and quality behavior of an alignment algorithm depending on several independent parameters.

The computation of all examples presented here was performed on the compute server of the Bielefeld Bioinformatics WebServer, a 167-MHz Sun SparcStation Enterprise with 256 Mb of RAM and 512 Mb of swap space running Solaris 2.5.1.

3.1. *The general behavior of DCA*

Except when stated otherwise, the sequences used in the following are simulated 'proteins' with an average pairwise sequence similarity of 250 PAM. The expected average length is 250 letters, and the size of the families ranges from $k=3$ to $k=14$ sequences. The recursion stop size of DCA is set to $L=40$. Since the sequences of the example families considered here are rather equally distributed, we use the unweighted SP score, i.e. all weights are set to $\alpha_{p,q}=1$. All results presented in this section are average values over 100 runs with different sequence families.

The interdependence of DCA's computation time on the one hand and the alignment quality on the other hand, depending on the recursion stop size L, has already been shown in Tönges et al. (1996). Fig. 4 (left hand side: average score error, i.e. the relative difference of the score of an alignment computed by DCA and that of a score-optimal alignment computed by MSA; right hand side: average computation time; note the logarithmic time scale) shows similar results for a larger parameter space of three up to six sequences. Although DCA could compute alignments for even much larger sequence families (as will be shown below), it was not possible to obtain optimal alignments with MSA for all 100 families with seven and more sequences, which we needed for the comparison of alignment scores. The quality versus time trade-off, which is discussed in detail by Tönges et al. (1996), is confirmed. For the small sequence families used here, a value for L of between 40 and 100 seems a good compromise with a rather high alignment quality and still comparatively quick computation times. For larger sequence families, of course, a smaller value for L between 20 and 40 should be preferred.

It is noteworthy that the values for $L=20$ and $L=30$, and the values for $L=40$ and $L=50$ almost always coincide. This can be easily understood by observing the series of average sequence lengths of the subsequences when starting with an initial length of $n=250$: 125, 63, 31, 16,... Both $L=20$ and $L=30$ (as well as $L=40$ and $L=50$) fall into the same class, and thus they have (in most cases) the same number of recursions, resulting in the same subsequence families aligned by MSA.

Fig. 5 shows the computation time of DCA for different sequence lengths. The curves show a quadratic behavior that can also be theoretically devised (Stoye, 1997).

The corresponding memory usage is shown in Fig. 6. While, in the theoretical worst case, the memory requirement of DCA grows quadratically with the number and length of the sequences (Stoye, 1997), the practical increase of memory usage with sequence length seems to be almost linear (the non-monotonicities are due to boundary effects for short sequences).

The time and memory requirements of DCA depending on the number of sequences are shown in Fig. 7. Up to 11 of our random sequences of length $n=600$ can be aligned within less than half a minute of computation time.

Finally, we have evaluated the dependence of DCA on the similarity of the sequences. We have created random sequence families with average similarities ranging from 100 up to 1000 PAM. Again, the sequences are of an average expected length of 250. Time and memory usage of DCA are shown in Fig. 8. As is also true for other alignment programs, the closer the sequences are related, the faster the algorithm proceeds and—due to the better behavior of our speed-up heuristics—the less memory is consumed.

3.2. *Four benchmark families*

McClure et al. (1994) applied a variety of multiple alignment programs to four protein families covering a wide range of sequence divergence: 12 globins, 12 kinases, 12 aspartic acid proteases, and 12 ribonuclease H (RH) sequences, respectively. They also defined sub-

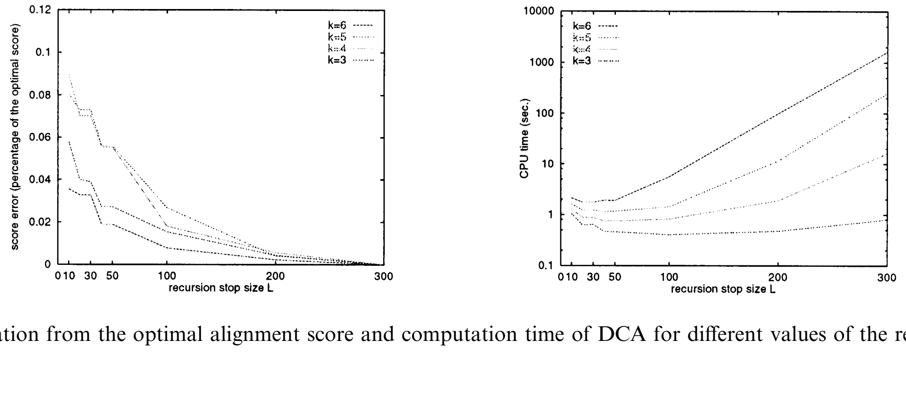

Fig. 4. Relative deviation from the optimal alignment score and computation time of DCA for different values of the recursion stop size, L.

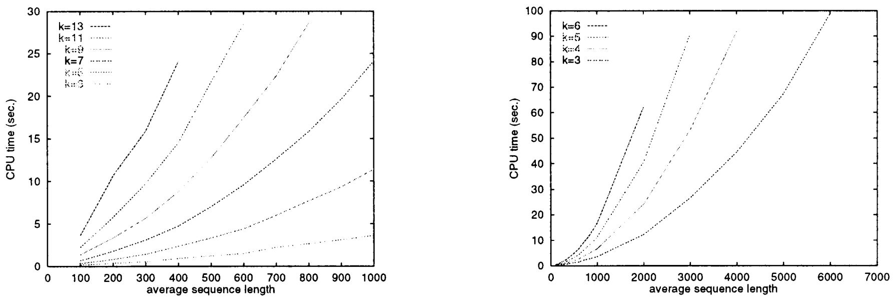

Fig. 5. Time usage of DCA versus sequence length.

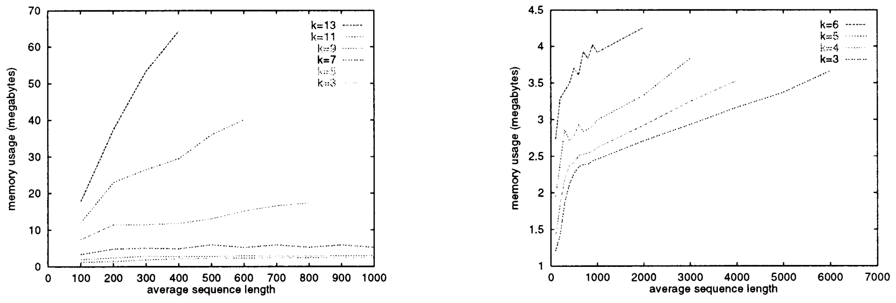

Fig. 6. Memory usage of DCA versus sequence length.

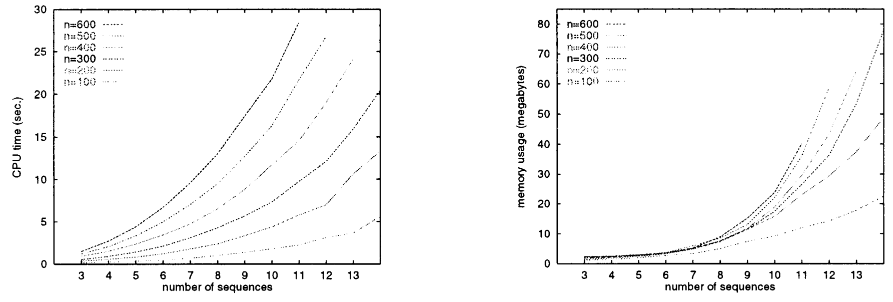

Fig. 7. Time and memory usage of DCA versus number of sequences.

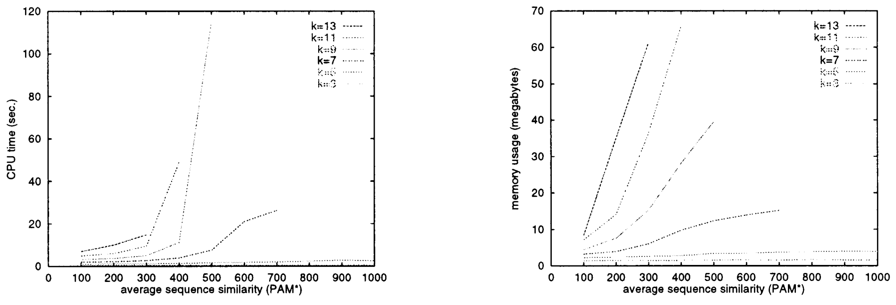

Fig. 8. Time and memory usage of DCA versus sequence similarity.

families containing the six and 10 sequences of each family with the widest distance distribution of sequence relationship (see Table 1).

Whereas the globins and the kinases are rather similar and hence the computation of reasonable alignments of these sequences is not difficult, the protease and RH sequences are much more diverse. Here, several of the tested alignment programs performed less well, and some of the programs could even not align these sequences at all. The fragment-based method ASSEMBLE (Vingron and Argos, 1991)—which produces excellent alignments of the globins and the kinases—for example had enormous problems in detecting reliable anchor subsequences in the protease sequences and the RH proteins (McClure et al., 1994).

The output of the alignment programs was scored by

Table 1
The sequences used in the study of McClure et al. (1994)

Globins 6	HAHU, HBHU, MYHU, IGLOB, HEYL, HEMB
Globins 10	HAHU, HADK, HBHU, HBDK, MYHU, MYOR, IGLOB, HENL, HEYL, HEMB
Globins 12	HAHU, HAOR, HADK, HBHU, HBOR, HBDK, MYHU, MYOR, IGLOB, HENL, HEYL, HEMB
Kiniases 6	CAPK, CD28, WEE1, VFES, PDGM, EGFR
Kinases 10	CAPK, PSKH, CD28, WEE1, RAF1, CMOS, VFES, PDGM, EGFR, HSVK
Kinases 12	CAPK, MLCK, PSKH, CD28, WEE1, RAF1, CMOS, CSRC, VFES, PDGM, EGFR, HSVK
Proteases 6	MoMLV, CaMV, 17.6, TY 3, COPIA, PEPH
Proteases 10	HTLVI, RSV, HIVI, MoMLV, CaMV, TY 3, COPIA, PEPH, PECH, PEPP
Proteases 12	HTLVI, RSV, HIVI, SRV-I, MoMLV, CaMV, 17.6, TY 3, COPIA, PEPH, PECH, PEPP
RH 6	HTL2, ROUS, MMLV, 176H, HEPB, ECOL
RH 10	HTL2, SRVI, ROUS, HIV2, MMLV, INGT, CAMV, 176H, COPH, ECOL
RH 12	HTL2, SRVI, ROUS, HIV2, MMLV, INGT, CAMV, 176H, MAUP, HEPB, COPH, ECOL

The sequence names are linked directly to the corresponding PIR (for the globins) and SwissProt entries of the NCBI Entrez database browser. Note that in the DCA runs described below, we align exactly the sequences used by McClure et al. (1994), which, in some cases, differ slightly from the sequences in the databases.

the following procedure: From structurally verified alignments of the test families, highly conserved regions—so-called sequence motifs—of three to nine amino acids and some single completely conserved residues (for convenience, also called motifs) were extracted: five motifs in the globins family, eight in the kinase, three in the protease, and four motifs in the RH family. Then—individually for each motif—the percentage of the number of sequences in each data set was measured, for which the motif was correctly identified (i.e. all positions of the motif coincide). If a motif was aligned correctly in more than one subfamily of the sequences without aligning these blocks to one another, the total percentage correct match was a combined score of the aligned subfamilies.

Also, a condensed way of presenting the results has been used (Gupta et al., 1995): the scores of all individual motifs are added, and the sum is divided by 100. When motifs are spread over more than one subfamily of the aligned sequences, we will indicate this by an asterisk. Thus, a single number gives an impression of the quality of an alignment. In the tables below, the individual scores of the distinct motifs as well as the complete alignments computed by DCA are displayed upon clicking on the corresponding score values.

We have run DCA with different values of the recursion stop size, L, and different substitution matrices: two matrices from the PAM seriers (Dayhoff et al., 1979) and three matrices from the Blosum series (Henikoff and Henikoff, 1992). We converted these matrices to distance (rather than similarity) scores and then—due to requirements of MSA—shifted them to non-negative values. For each substitution matrix, we fixed the gap parameters to values that seemed to yield the best results. Table 2 lists the ranges of the entries in the used substitution matrices as well as the corresponding gap functions. By clicking on their names, the full matrices are displayed.

Table 3 shows the results and computation times of DCA for $L=20$, and Table 4 shows the results for $L=40$. Some of the runs took rather a long time (more than 50 h of CPU time) and have therefore been stopped. This is indicated by a question mark.

The globins are very rapidly aligned by DCA, and the results are almost always close-to-optimal or optimal. Also for the kinases (which are comparatively long protein sequences) and for the smaller families of the kinase and RH sequences, DCA is relatively fast. However, the families of 12 protease and 12 RH sequences require extensively more time. Neither the length nor the number of sequences seem to have the highest influence on the computation time of DCA, but rather the sequence similarity has. In addition, some of the families with 10 sequences take, for $L=40$, considerably longer to compute than for $L=20$ indicating long MSA runs of the comparatively long subsequences of length ≤ 40.

An influence of the score function on the computation time is also observed. Some alignments with the PAM 160 matrix take more than 50 times as long as the corresponding runs with the Blosum 62 matrix. This is due to the high influence of the chosen substitution matrix and gap function on the effectiveness of our method for speeding up the search for good cut positions (Stoye, 1997).

We have also developed a heuristic method allowing large amounts of computation time to be skipped in our optimization procedure, with the drawback of slightly less accurate, so-called approximate cut positions. Here, the cut positions are computed by an iterated greedy procedure which was originally developed to speed-up the standard DCA method (Perrey and Stoye, 1996). The results obtained with this procedure and with $L=20$ are shown in Table 5.

Compared to Table 3, the computation times of the

Table 2
The substitution matrices and corresponding gap functions used in this study

Substitution matrix	Lowest distance	Highest distance	Gap function
PAM 250	0	25	$g(l)=8+12l$
PAM 160	0	29	$g(l)=8+12l$
Blosum 62	0	15	$g(l)=6+10l$
Blosum 45	0	20	$g(l)=10+9l$
Blosum 30	0	27	$g(l)=10+11l$

Table 3
Score and computation time of DCA with $L = 20$ using different amino acid substitution matrices

Sequences	Motifs	PAM 250		PAM 160		Blosum 62		Blosum 45		Blosum 30	
Globins 6	5.00	4.83	0.9 s	4.83	1.0 s	5.00	0.8 s	4.83	1.2 s	4.67	1.3 s
Globins 10	5.00	4.90	2.7 s	4.90	3.0 s	5.00	2.7 s	4.90	3.3 s	*4.90	3.6 s
Globins 12	5.00	4.92	4.9 s	4.92	5.4 s	5.00	4.1 s	4.92	5.0 s	4.83	6.1 s
Kinases 6	8.00	7.67	4.9 s	*8.00	3.9 s	7.83	4.3 s	8.00	4.0 s	7.67	3.9 s
Kinases 10	8.00	7.70	2.9 min	7.70	13.9 min	7.80	1.5 min	7.90	26.2 min	8.00	3.3 h
Kinases 12	8.00	7.83	6.5 min	8.00	28.2 min	7.92	2.8 min	7.92	18.0 min	8.00	14.6 min
Proteases 6	3.00	*1.83	1.9 s	1.33	5.8 s	1.00	1.5 s	1.67	5.0 s	2.17	10.3 s
Proteases 10	3.00	*2.00	4.1 min	*2.40	14.9 min	*2.20	7.2 min	*2.20	38.1	*2.20	2.2 h
Proteases 12	3.00	*2.25	1.5 h	?		*2.00	19.2 h	?		?	
RH 6	4.00	2.67	1.4 s	*3.83	1.6 s	3.00	1.1 s	3.33	1.8 s	3.67	2.0 s
RH 10	4.00	*3.50	1.7 min	*3.50	8.5 min	*3.30	25.4 s	*3.70	12.9 min	3.60	63.6 min
RH12	4.00	*2.83	28.2 min	?		*3.42	2.2	?		?	

Table 4
Score and computation time of DCA with $L = 40$ using different amino acid substitution matrices

Sequences	Motifs	PAM 250		PAM 160		Blosum 62		Blosum 45		Blosum 30	
Globins 6	5.00	4.83	0.9 s	4.67	0.9 s	4.83	0.9 s	4.83	1.1 s	4.67	1.0 s
Globins 10	5.00	4.90	2.8 s	5.00	13.6 s	5.00	3.3 s	4.90	3.3 s	4.80	3.4 s
Globins 12	5.00	4.92	5.0 s	4.83	5.0 s	5.00	4.1 s	4.92	5.1 s	4.83	5.6 s
Kinases 6	8.00	7.83	4.5 s	8.00	4.7 s	7.83	4.5 s	7.83	4.1 s	7.67	4.0 s
Kinases 10	8.00	7.70	3.1 min	7.80	32.4 min	*7.70	1.6 min	7.90	26.3 min	7.90	3.4 h
Kinases 12	8.00	7.67	7.3 min	7.92	32.7 min	7.92	2.8 min	7.92	21.2 min	8.00	1.3 h
Proteases 6	3.00	*1.83	33.2 s	1.33	17.3 min	1.33	3.4 s	1.83	32.9 s	*2.50	45.8 min
Proteases 10	3.00	*2.10	4.0 min	*2.40	8.0 h	1.90	6.6 min	*2.40	3.8 h	*2.30	4.2 h
Proteases 12	3.00	*2.17	1.5 h	?		*2.08	19.0h	?		?	
RH 6	4.00	*3.00	1.5 s	3.33	3.7 s	3.17	1.1 s	*3.50	3.2 s	3.67	3.0 min
RH 10	4.00	*3.60	3.1 min	*3.50	2.6 h	*3.30	1.2 h	3.50	18.4 min	3.50	1.5 h
RH 12	4.00	*3.25	27.7 min	?		*3.42	2.2 h	?		?	

Table 5
Score and computation time of DCA with $L = 20$ when approximate cut positions are used

Sequences	Motifs	PAM 250		PAM 160		Blosum 62		Blosum 45		Blosum 30	
Globins 6	5.00	4.83	0.9 s	4.83	1.0 s	*4.83	0.8 s	*4.67	1.0 s	*4.67	1.3 s
Globins 10	5.00	5.00	2.7 s	4.80	3.1 s	5.00	2.5 s	4.90	2.9 s	5.00	3.2 s
Globins 12	5.00	5.00	4.3 s	4.83	4.0 s	4.92	4.5 s	4.92	4.6 s	4.83	5.3 s
Kinases 6	8.00	7.50	4.0 s	7.50	3.4 s	7.67	4.0 s	7.50	3.4 s	7.33	3.5 s
Kinases 10	8.00	7.40	15.0 s	7.50	13.3 s	7.80	12.9 s	7.90	13.7 s	8.00	16.5 s
Kinases 12	8.00	7.92	19.6 s	*7.83	19.7 s	7.83	18.7 s	8.00	20.5 s	8.00	21.5 s
Proteases 6	3.00	1.50	1.1 s	1.33	1.7 s	0.67	1.0 s	1.83	1.2 s	*1.83	1.5 s
Proteases 10	3.00	*2.10	8.2 s	*2.30	3.4 s	*1.90	3.5 s	*2.40	3.1 s	*2.50	13.7 s
Proteases 12	3.00	*2.17	4.6 s	*2.25	67.1 s	*1.92	4.5 s	*2.33	5.6 s	*2.50	2.4 min
RH 6	4.00	2.50	1.2 s	2.67	1.1 s	2.83	1.0 s	3.50	1.2 s	3.33	1.6 s
RH 10	4.00	*3.10	3.7 s	*3.60	3.9 s	*3.50	3.6 s	3.40	3.9 s	3.50	3.9 s
RH 12	4.00	*3.25	5.5 s	*3.33	6.5 s	*3.00	5.9 s	3.25	5.8 s	*3.42	6.0 s

larger sequence families are reduced enormously when approximate cut positions are used (see Table 5). Each of the sequence families can be aligned within several seconds up to slightly above 1 min. Where the computation of ordinary cut positions takes extremely long (e.g. for the family of twelve protease sequences), a speed-up factor of more than 1000 is achieved. Accompanied with this speed increase, only a low decrease of alignment accuracy is observed. Often, the same number of motifs are aligned correctly. Occasionally, the score even increases (e.g. for the 12 kinases with the Blosum 45 substitution matrix).

In general, we have observed that substitution matrices from the Blosum series on these data produce slightly better results than the corresponding PAM matrices. [Due to Henikoff and Henikoff (1992), the PAM 250 matrix is comparable to Blosum 45, and PAM 160 is comparable to Blosum 62.] This result is in accordance with Henikoff and Henikoff (1993), who also observed that the Blosum matrices perform better for distantly related proteins.

In Table 6, we compare the best alignments obtained with DCA to the results of the alignment programs DFALIGN (Feng and Doolittle, 1987) and AMULT (Barton and Sternberg, 1987a,b), which were the best and second best scoring programs in the study of McClure et al. Both DFALIGN and AMULT are implementations of the progressive sequence alignment approach. McClure et al. do not give the computation times of the methods that they tested. Therefore, we can only compare the quality of the alignments. DCA outperforms AMULT in all cases and produces results comparable to those of DFALIGN. In four cases, DCA computes alignments scoring higher than any of the programs evaluated in the study of McClure et al. (1994). This proves that—provided the score function is selected carefully—the divide-and-conquer alignment method can compete with the best alignment programs currently available.

3.3. *Assessment of alignment score functions*

Given the results of the previous section, of course, we wondered why, for some sequence families, the results obtained with DCA are still slightly different from the biologically correct alignments despite the great proximity of our alignments to the SP optimal ones. Also, of course, the answer is that our alignments can hardly be better than the score function that we approximate. Consequently, we have compared the score of alignments computed with DCA to that of the biologically correct 'true' alignments as published in McClure et al. (1994). The result of this comparison is presented in Tables 7 and 8. For the example of the PAM 250 score, Table 7 explains how we compute the relative difference of the score of the DCA-alignment and the score of the true alignment. Table 8 shows the relative differences for all the examined sequence families and substitution matrices.

In all cases, the score of an alignment computed with

Table 7
Comparison of the absolute PAM 250 scores of the true alignments and of those computed with DCA ($L=20$)

Sequences	True	DCA	Difference	Relative difference
Globins 6	37 054	36 834	220	0.60%
Globins 10	108 460	108 093	367	0.34%
Globins 12	156 074	155 657	417	0.27%
Kinases 6	73 685	71 249	2436	3.42%
Kinases 10	217 760	214 661	3099	1.44%
Kinases 12	314 288	308 662	5626	1.82%
Proteases 6	36 089	34 138	1951	5.71%
Proteases 10	107 085	103 972	3113	2.99%
Proteases 12	156 051	151 663	4388	2.89%
RH 6	40 334	37 596	2738	7.28%
RH 10	118 720	112 129	6591	5.88%
RH 12	178 069	168 600	9469	5.62%

The relative difference is the absolute difference divided by the score of the DCA alignment.

Table 6
Numbers of correctly aligned motifs in alignments computed with the programs DFALIGN and AMULT compared to the highest scoring alignments computed with DCA

Sequences	Motifs	DFALIGN	AMULT	DCA	(DCA score function)
Globins 6	5.00	5.00	5.00	5.00	Blosum 62, $L=20$
Globins 10	5.00	5.00	5.00	5.00	e.g. Blosum 62, $L=20$
Globins 12	5.00	5.00	5.00	5.00	e.g. Blosum 62, $L=20$
Kinases 6	8.00	7.67	7.33	8.00	e.g. Blosum 45, $L=20$
Kinases 10	8.00	8.00	7.70	8.00	e.g. Blosum 30, $L=20$
Kinases 12	8.00	8.00	7.75	8.00	e.g. Blosum 30, $L=20$
Proteases 6	3.00	2.33	1.17	*2.50	Blosum 30, $L=40$
Proteases 10	3.00	*3.00	*2.40	*2.50	Blosum 30, $L=20$, approx.
Proteases 12	3.00	*3.00	*2.40	*2.50	Blosum 30, $L=20$, approx.
RH 6	4.00	3.67	*3.30	*3.83	PAM 160, $L=20$
RH 10	4.00	3.30	3.20	*3.70	Blosum 45, $L=20$
RH 12	4.00	3.83	*2.92	*3.42	e.g. Blosum 30, $L=20$, approx.

Table 8
Relative difference of scores of the true alignments and of those computed with DCA ($L=20$) for the different amino acid substitution matrices

Sequences	PAM 250	PAM 160	Blosum 62	Blosum 45	Blosum 30
Globins 6	0.60%	0.75%	0.93%	0.61%	0.45%
Globins 10	0.34%	0.45%	0.68%	0.38%	0.08%
Globins 12	0.27%	0.24%	0.60%	0.11%	0.33%
Kinases 6	3.42%	3.30%	4.63%	2.90%	1.99%
Kinases 10	1.44%	0.72%	3.44%	1.15%	0.16%
Kinases 12	1.82%	0.97%	3.70%	1.81%	0.74%
Proteases 6	5.71%	4.76%	8.78%	3.44%	2.41%
Proteases 10	2.99%	1.19%	5.04%	0.88%	0.24%
Proteases 12	2.89%	?	3.82%	?	?
RH 6	7.28%	5.58%	10.69%	6.66%	4.54%
RH 10	5.88%	3.45%	8.11%	3.91%	3.18%
RH 12	5.62%	?	8.78%	?	?

DCA is lower than that of the corresponding true alignment. However, for the globins and the kinases—where we detected almost all motifs correctly—both scores differ much less than for the proteases and the RH proteins. It also can be observed that the subfamilies of six sequences are much harder to align than the larger families, which is in accordance with our results shown in Tables 3–5. Assuming that the score of an alignment computed with DCA differs by less than 1% from the optimal score, this proves that the studied alignment score functions—even if we could compute an SP optimal alignment—will not allow a biologically correct alignment of the RH sequences, for example, to be computed. To close this gap, further work on the development of better alignment score functions will be necessary.

Similar to the results shown in the previous section, this comparison of alignment scores shows that the alignments computed with the Blosum matrices (in particular Blosum 45 and Blosum 30) are mostly closer to the true alignments than those computed with the matrices from the PAM series. With this study, we have shown that due to its speed and high accuracy of the results, DCA makes it possible to analyze directly the properties of multiple alignment score functions.

3.4. Comparison with MSA

The authors of the improved version 2.0 of MSA, Gupta et al. (1995), applied their alignment program to the same sequences as those used in the comparison of McClure et al. described above. Because they could still not align the full data sets, they selected some subfamilies (denoted by the letters A, B, C) that MSA was able to align SP-optimally [with regard to PAM 250 and gap function $g(l)=8+12l$].

In Table 9, we report the results of Gupta et al. (1995) compared to the results of DCA on the same subfamilies. For better comparability of our computation times to those of MSA reported in the study by Gupta et al., we used for these runs a Sun SparcStation 10 as they did in their study.

The speed-up factor of DCA over MSA ranges from 12.8 to over 1100, and the memory usage of DCA is two to 20 times lower than that of MSA. Moreover, our alignments with the same substitution matrix often find the same number of motifs as those computed with MSA. In four cases, there are less, and in one case, even more motifs are aligned correctly. Again, with matrices from the Blosum series, the results can be improved. For all sequence families, DCA can compute alignments that score higher than, or equal to, the SP-optimal one regarding the PAM 250 score. This again supports our assertion that the alignment score function influences the alignment quality (in biological terms) much more than the remaining difference of less than 1% between an alignment computed with DCA and an SP-optimal one.

4. Conclusions

Due to the generalizations described, the divide-and-conquer algorithm for an approximate solution of the global multiple sequence alignment problem is now applicable to real-world alignment tasks. Experimental results indicate that the computed alignments are comparable to those of other state-of-the-art alignment programs. Furthermore, since the alignment is simultaneous, i.e. not based on a pre-given or pre-computed alignment guide tree, the alignments are well suited also as an unbiased starting point for the reconstruction of evolutionary relationships.

The basic DCA algorithm is quite simple: The main parameter, the recursion stop size, L, is easily understood and allows a high degree of control over the performance of the program. This might be an important step for multiple sequence alignment from being a black box for molecular biologists toward becoming a mechanism with transparent behavior and performance.

Table 9
Running time and percent correctly aligned motifs in alignments computed with MSA (using the PAM 250 substitution matrix) and the corresponding values of DCA (PAM 250 and the best-scoring matrix from the Blosum series)

Sequences	Number	(Length)	Motifs	MSA	(PAM250)	DCA	(PAM250)	DCA	(Blosum)
Globins A	7	(141–153)	5.00	4.86	157 s	4.86	4.4 s	5.00	5.4 s
Globins B	10	(141–153)	5.00	5.00	130 s	4.90	10.1 s	5.00	10.5 s
Kinases A	5	(255–293)	8.00	8.00	10 min	8.00	7.9 s	8.00	17.2 s
Kinases B	6	(255–293)	8.00	8.00	118 min	8.00	9.7 s	8.00	61.8 s
Kinases C	4	(255–339)	8.00	6.75	210 s	*7.50	4.6 s	7.25	4.9 s
Proteases A	5	(998–150)	3.00	2.80	37 s	2.40	2.5 s	2.80	19.7 s
Proteases B	4	(113–150)	3.00	0.50	9 min	0.00	1.5 s	1.00	3.7 s
RH A	5	(126–157)	4.00	2.60	68 min	*2.60	3.5 s	3.40	32.1 s

Due to its simplicity, the algorithm is also highly suitable for incorporation into larger systems that require a number of reliable, but not necessarily optimal multiple sequence alignments. The version of DCA for three sequences has already been incorporated in a program that simultaneously computes an alignment and reconstructs a phylogenetic tree (Bergmann et al., in preparation). For a generalization of this method, we also plan to use the general DCA algorithm presented here.

Future work also seems valuable in the following direction. As noted in the Introduction, the divide-and-conquer alignment procedure is related to fragment-based multiple alignment methods, yet using a more systematic way of computing the anchor positions. However, if there are obvious anchors obtainable from local alignments, the effort in computing cut positions can be circumvented. A combination of both approaches may be a suitable solution. Anchors are computed as in one of the standard fragment-based methods (e.g. Schuler et al., 1991; Vingron and Argos, 1991; Morgenstern et al., 1996) demanding high similarity scores if necessary, and cut positions are computed between them if—due to the high requirements for fragment similarity—the intermediate regions are too large. Then, the remaining (short) subsequences in between are aligned optimally as in the standard algorithms. Following this outline, it should be possible to develop an algorithm that is faster and produces more accurate alignments than any of the separate approaches.

Beyond the evaluation of the divide-and-conquer algorithm, our studies in biological sequence data have raised an important aspect of multiple sequence alignment in practice. The relatedness of the sequences can have a much greater influence on computation time than the length and/or the number of sequences. Whereas 40 highly similar cytochrome C sequences can be aligned by DCA in less than a minute (Stoye, 1997), the alignment of 12 less related protease sequences of about the same length takes more than an hour of computation time.

Finally, we believe that with DCA, we have reached a limit of what can be done with the SP model and the commonly used alignment score functions. For obtaining results that are still nearer to biologically correct alignments, it seems that more sophisticated score functions incorporating further biological criteria have to be considered.

Acknowledgement

I am very grateful to Andreas Dress and Sören Perrey for fruitful discussions during the preparation of this paper. I would also like to thank Marcella McClure for providing me with the sequences discussed in the Results. This work was supported by the German Ministry for Research and Technology (BMBF) under grant number 01 IB 301 B4, by the Graduate College Strukturbildungsprozesse (Bielefeld), and by the German Academic Exchange Service (DAAD).

References

Altschul, S.F., Erickson, B.W., 1986. Optimal sequence alignment using affine gap costs. Bull. Math. Biol. 48 (5/6), 603–616.

Altschul, S.F., Lipman, D.J., 1989. Trees, stars, and multiple biological sequence alignment. SIAM J. Appl. Math. 49 (1), 197–209.

Argos, P., Vingron, M., Vogt, G., 1991. Protein sequence comparison: Methods and significance. Prot. Eng. 4 (4), 375–383.

Barton, G.J., Sternberg, M.J.E., 1987a. A strategy for the rapid multiple alignment of protein sequences—confidence levels from tertiary structure comparisons. J. Mol. Biol. 198, 327–337.

Barton, G.J., Sternberg, M.J.E., 1987b. Evaluation and improvements in the automatic alignment of protein sequences. Prot. Eng. 1 (2), 89–94.

Ben-Dor, A., Lancia, G., Perone, J., Ravi, R., 1997. Banishing bias from consensus sequences. In: Apostolico, A., Hein, J. (Eds.), Combinatorial Pattern Matching: 8th Annual Symposium, CPM 97. Aarhus, Denmark, June/July 1997. Proc. No. 1264 in Lecture Notes in Computer Science. Springer, Berlin, pp. 247–261.

Bergmann, K., Dress, A., Stoye, J., Vingron, M., in preparation. A practical approach to integrated alignment and phylogeny construction.

Carrillo, H., Lipman, D., 1988. The multiple sequence alignment problem in biology. SIAM J. Appl. Math. 48 (5), 1073–1082.

Chan, S.C., Wong, A.K.C., Chiu, D.K.Y., 1992. A survey of multiple sequence comparison methods. Bull. Math. Biol. 54 (4), 563–598.

Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C., 1979. A model of evolutionary change in proteins. In Dayhoff, M.O. (Ed.), Atlas of Protein Sequence and Structure, Vol. 5, Suppl. 3. National Biomedical Research Foundation, Washington, DC, pp. 345–352.

Feng, D.-F., Doolittle, R.F., 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. 25, 351–360.

Gotoh, O., 1982. An improved algorithm for matching biological sequences. J. Mol. Biol. 162, 705–708.

Gotoh, O., 1996. Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments. J. Mol. Biol. 264, 823–838.

Gupta, S.K., Kececioglu, J.D., Schäffer, A.A., 1995. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. J. Comp. Biol. 2 (3), 459–472.

Gusfield, D., 1993. Efficient methods for multiple sequence alignment with guaranteed error bounds. Bull. Math. Biol. 55 (1), 141–154.

Hein, J., 1994. TreeAlign. In: Griffin, A.M., Griffin, H.G. (Eds.), Computer Analysis of Sequence Data, Part II, Vol. 25 of Methods in Molecular Biology. Humana Press, Totowa, NJ, 1994, pp. 349–364.

Henikoff, S., Henikoff, J.G., 1992. Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA 89, 10915–10919.

Henikoff, S., Henikoff, J.G., 1993. Performance evaluation of amino acid substitution matrices. Proteins 17, 49–61.

Lake, J.A., 1991. The order of sequence alignment can bias the selection of tree topology. Mol. Biol. Evol. 8 (3), 378–385.

Lipman, D.J., Altschul, S.F., Kececioglu, J.D., 1989. A tool for multiple sequence alignment. Proc. Natl. Acad. Sci. USA 86, 4412–4415.

McClure, M.A., Vasi, T.K., Fitch, W.M., 1994. Comparative analysis of multiple protein-sequence alignment methods. Mol. Biol. Evol. 11 (4), 571–592.

Morgenstern, B., Dress, A.W.M., Werner, T., 1996. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. Proc. Natl. Acad. Sci. USA 93 (22), 12098–12103.

Needleman, S.B., Wunsch, C.D., 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453.

Perrey, S.W. and Stoye, J., 1996. Fast approximation to the NP-hard problem of multiple sequence alignment. Information and Mathematical Science Reports, Series B:96/06 (ISSN 1171-7637).

Pevzner, P.A., 1992. Multiple alignment with guaranteed error bounds and communication cost. In: Apostolico, A., Crochemore, M., Galil, Z., Manber, U. (Eds.), Combinational Pattern Matching: Third Annual Symposium, CPM 92. Tucson, AZ, April/May 1992. Proc. No. 644 in Lecture Notes in Computer Science. Springer, Berlin, pp. 205–213.

Schuler, G.D., Altschul, S.F., Lipman, D.J., 1991. A workbench for multiple alignment construction and analysis. Proteins 9, 180–190.

Stoye, J., 1997. Divide-and-Conquer Multiple Sequence Alignment. Dissertation, Technische Fakultät der Universität Bielefeld. Available online from http://www.mathematik.uni-bielefeld.de/~stoye/preprints/report97-02.ps.gz.

Stoye, J., Perrey, S.W., Dress, A.W.M., 1997a. Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment. Appl. Math. Lett. 10 (2), 67–73.

Stoye, J., Moulton, V., Dress, A.W.M., 1997b. DCA: An efficient implementation of the divide-and-conquer approach to simultaneous multiple sequence alignment. CABIOS 13 (6), 625–626.

Stoye, J., Evers, D., Meyer, F., in press. Rose: Generating sequence families. Bioinformatics 14(3).

Thorne, J.L., Kishino, H., 1992. Freeing phylogenies from artifacts of alignment. Mol. Biol. Evol. 9 (6), 1148–1162.

Tönges, U., Perrey, S.W., Stoye, J., Dress, A.W.M., 1996. A general method for fast multiple sequence alignment. Gene 172, GC33–GC41.

Vingron, M., Argos, P., 1990. Determination of reliable regions in protein sequence alignments. Prot. Eng. 3 (7), 565–569.

Vingron, M., Argos, P., 1991. Motif recognition and alignment for many sequences by comparison of dot-matrices. J. Mol. Biol. 218, 33–43.

Wang, L., Jiang, T., 1994. On the complexity of multiple sequence alignment. J. Comp. Biol. 1 (4), 337–348.

Waterman, M.S., 1983. Sequence alignments in the neighborhood of the optimum with general application to dynamic programming. Proc. Natl. Acad. Sci. USA 80, 3123–3124.