Abstract

Motivation: The performance and time complexity of an improved version of the segment-to-segment approach to multiple sequence alignment is discussed. In this approach, alignments are composed from gap-free segment pairs, and the score of an alignment is defined as the sum of so-called weights of these segment pairs.

Results: A modification of the weight function used in the original version of the alignment program DIALIGN has two important advantages: it can be applied to both globally and locally related sequence sets, and the running time of the program is considerably improved. The time complexity of the algorithm is discussed theoretically, and the program running time is reported for various test examples.

Availability: The program is available on-line at the Bielefeld University Bioinformatics Server (BiBiServ) http://bibiserv.TechFak.Uni-Bielefeld.DE/dialign/

Contact: burkhard.morgenstern@rp-rorer.co.uk

Introduction

Since the early 1970s, most methods for sequence alignment have relied on maximizing an alignment score that is defined as the sum of similarity values of matched individual residues minus a so-called gap penalty for every gap introduced into the alignment (Needleman and Wunsch, 1970). Frequently, the alignment problem is just seen as the problem of finding optimal alignments according to the definition of Needleman and Wunsch. Some authors deal with ‘optimal’ or ‘mathematically correct’ alignments without explicitly mentioning an underlying objective function—it is simply assumed that ‘optimality’ and ‘correctness’ always refer to the scoring scheme as defined by Needleman and Wunsch.

It is well known that alignment methods based on the Needleman–Wunsch scoring scheme are capable of producing biologically meaningful alignments as long as sequences are globally related and not too many gaps need to be inserted to align the related parts of the sequences. In these situations, high-scoring alignments in the sense of Needleman and Wunsch correspond to biologically correct alignments. This is due to the fact that optimal Needleman–Wunsch alignments maximize certain likelihood functions defined in terms of evolutionary events that may have occurred since sequences have evolved from a putative common ancestor (Bishop and Thompson, 1986; Thorne et al., 1992; Thorne and Churchill, 1995; Zhu et al., 1998).

On the other hand, there are many sequence families that share only isolated regions of local similarity. Here, it is not possible to express the degree of similarity or the distance among sequences by single evolutionary events. The question is not which or how many mutations may have occurred since sequences have separated, it is, instead, whether one can detect any similarity at all among them. Therefore, alignment methods relying on the Needleman–Wunsch scoring scheme are not very capable of dealing with distantly related sequences.

To cope with these situations, a number of local pairwise alignment methods have been developed (Smith and Waterman, 1981; Pearson and Lipman, 1988; Altschul et al., 1990). These methods try to maximize the Needleman–Wunsch alignment score locally and ignore the remaining parts of the sequences. In addition, there are various methods for local multiple alignment that construct alignments from blocks of gap-free segments of the sequences (Waterman, 1986; Depiereux and Feytmans, 1992; Lawrence et al., 1993; Depiereux et al., 1997). Limitations of these methods are that (i) every motif is required to occur in all sequences and (ii) all instances of a motif must have the same length.

Schuler et al. (1991) discussed this problem and proposed a local multiple alignment strategy without these limitations. More recently, Brocchieri and Karlin (1998) proposed another local alignment method: based on the significant
Fig. 1. Consistent collections of diagonals (segment pairs) and alignments in pairwise and multiple sequence comparison. If two sequences are to be aligned, a collection of diagonals is consistent if, and only if, diagonals are ordered as indicated in (A). In this case, there exists an alignment (B) such that all segment pairs are matched. Residues not involved in any of the selected diagonals are printed in lower-case letters and are not considered to be aligned. Correspondingly, (C) represents a consistent collection of diagonals and (D) the resulting alignment for a multiple sequence comparison.

segment pair alignment (SSPA) protocol (Karlin and Brocchieri, 1996), they construct alignments from blocks of segments that are not required to involve all sequences under consideration. Another novel approach to local multiple alignment has been proposed by Abdeddaïm (1997). Here, a greedy approach is used to construct multiple alignments from local pairwise alignments in the sense of Smith and Waterman (1981).

Recently, we have proposed an alternative method for local multiple alignment (Morgenstern et al., 1996, 1998a). Alignments are composed from gap-free pairs of sequence segments of equal length. Such segment pairs are referred to as diagonals since they would appear as diagonal runs in the respective pairwise comparison matrices.

Pairwise as well as multiple alignments are considered to be represented by collections of diagonals meeting a certain consistency criterion. In short, a collection of diagonals is called consistent if there exists an alignment such that all segment pairs are matched (see Figure 1). If two sequences are to be aligned, consistency is given if diagonals (segment pairs) are ordered in the sense that, for any two diagonals, the end positions of one of them are both smaller than the respective starting positions of the other one. In multiple alignment, the question of consistency is more complicated. For a mathematical treatment of this problem, see Morgenstern et al. (1996).

In order to find ‘good’ collections of diagonals, a so-called weight function is used that gives a weight score to any possible diagonal. Given such a weight function, the optimization problem is to find a consistent collection of diagonals with maximum sum of weights. Thus, the basic difference between this method and other local or global alignment methods is the underlying objective function which is based on similarities among whole segments of the sequences rather than on similarities among single residues. The central question in this context is how to define an appropriate weight function on the set of all possible diagonals. Note that our novel objective function for sequence alignment does not employ any gap penalty. A first implementation of our method was called DIALIGN 1 (for DIagonal ALIGNment).

While most multiple alignment methods are either purely global or purely local methods, DIALIGN is able to cope with a variety of different situations. As indicated in Figure 1, the program can find local similarities in a multiple sequence comparison even if these similarities involve only two sequences. Different regions of similarity involving different sequences can be combined to one single multiple alignment, and non-related regions between these regions
are ignored. However, if sequences are globally related, DIALIGN will return a full global alignment.

Time complexity

Given a weight function defined on the set of all diagonals that can be formed, a certain modification of the well-known dynamic programming technique (Needleman and Wunsch, 1970) can be applied to find optimal pairwise alignments in the sense of the above definition, i.e., collections of diagonals with maximum sum of weights (see Morgenstern et al., 1996). In short, dynamic programming is applied to segment pairs rather than to pairs of single residues [see also Zhang et al. (1994) for a comparable approach]. Since there are \( O(L^3) \) possible diagonals in a comparison matrix, where \( L \) is the maximum length of the sequences, an optimal alignment can be found in \( O(L^5) \) time. If the length of diagonals under consideration is restricted, the running time of the pairwise alignment can be reduced to \( O(L^2) \). In practice, this restriction affects the quality of the resulting alignments only marginally since longer regions of gap-free similarity can be described by several consecutive diagonals instead of by a single one.

If \( N > 2 \) sequences are to be aligned, the DIALIGN program employs a greedy strategy (see Figure 2). First, all optimal pairwise segment-to-segment alignments are calculated. The set of all diagonals these respective pairwise alignments consist of is denoted by \( \mathcal{D}_1 \). Since, in general, the set \( \mathcal{D}_1 \) cannot be expected to be consistent, a consistent subset \( \mathcal{D}_2 \subset \mathcal{D}_1 \) is determined as follows. All diagonals contained in \( \mathcal{D}_1 \) are sorted according to their so-called overlap-weight scores. [The overlap weight of a diagonal reflects its weight as well as the degree of overlap with other diagonals in order to favor motifs occurring in more than two sequences, see Morgenstern et al. (1996). For example, in Figure 3, diagonals \( D_2 \) and \( D_4 \) have increased overlap weights compared to their original weights, whereas the overlap weights of the isolated diagonals \( D_1 \) and \( D_3 \) are not higher than the respective original weights.] Then, starting with the diagonal with maximum overlap weight, diagonals contained in \( \mathcal{D}_1 \) are included one by one in the set \( \mathcal{D}_2 \) provided they are consistent with the diagonals already included. Non-consistent diagonals are discarded (see Figure 3). This procedure is repeated iteratively until no additional diagonals can be found. Normally, this is the case after at most three iteration steps.

To decide whether a diagonal is consistent with the diagonals already included in the growing set \( \mathcal{D}_2 \), so-called consistency bounds \( b(x, i) \) and \( \overline{b}(x, i) \) have to be stored and updated. For a residue \( x \), \( b(x, i) \) denotes the position of the left-most residue within the \( i \)th sequence that can be aligned with \( x \) without causing inconsistencies with the diagonals already included in \( \mathcal{D}_2 \). \( \overline{b}(x, i) \) is the position of the right-most residue with this property [see Figure 4 and Morgenstern et al. (1996) for more details]. Similar ideas were proposed independently by Abbedaïm (1997). With every diagonal included in \( \mathcal{D}_2 \), the consistency bounds must be updated. It is easy to see that this can be done in \( O(N^2 \times L) \) time for every diagonal included in the set \( \mathcal{D}_2 \).

Every single iteration step of the algorithm involves four distinct procedures: (a) performing all optimal pairwise alignments; (b) calculating overlap weights for all diagonals contained in one of the optimal pairwise alignments; (c) sorting these diagonals; (d) including consistent diagonals in the set \( \mathcal{D}_2 \) and updating the consistency bounds \( b(x, i) \) and \( \overline{b}(x, i) \). There are \( O(N^2) \) optimal pairwise alignments to be calculated. If we denote the average number of diagonals in these pairwise alignments by \( n_a \), the set \( \mathcal{D}_2 \) consists of \( O(N^2 \times n_a) \) diagonals. Calculating the overlap weights requires all-against-all comparison of these diagonals, so this step takes \( O(N^4 \times n_a^2) \) time. Thus, the time complexity of one iteration step in our alignment procedure is:

\[
|N^2 \times L^2| + |N^2 \times n_a^2| + |N^2 \times n_a \times \log(N^2 \times n_a)| + |N^2 \times n_a \times N^2 \times L|
\]

(1)

In a final step, gaps are inserted in the sequences until all diagonals contained in \( \mathcal{D}_2 \) are matched. This can be done in \( N^2 \times L^2 \) time. Therefore, and since the number of iteration steps can be treated as a constant, equation (1) also describes the time complexity of the entire alignment procedure. An important consequence is that the program running crucially depends on the average number, \( n_a \), of diagonals in the optimal pairwise alignments.

Remark: It follows from equation (1) that the time complexity of the algorithm in terms of \( N \) and \( L \) is \( O(N^5 \times L^2) \).

Weight functions for diagonals

The weight function \( w_1 \) used in the first version of the DIALIGN program (DIALIGN 1) is based on an idea proposed by Altschul and Erickson (1986). Given a diagonal \( D \) of length \( l_D \), we denote by \( S_D \) the sum of the individual similarity values of residue pairs within this diagonal. If protein sequences are to be aligned, one of the usual substitution matrices, e.g. BLOSUM62 (Henikoff and Henikoff, 1992), may be used. For DNA sequences, similarity values are 1 for matches and 0 for mismatches.

Next, by \( P_1(l_D, S_D) \) we denote the probability that a random diagonal of the same length \( l_D \) has at least the same sum \( S_D \) of similarity values. Mathematically, this probability is given as a sum of convolution products of the probability distribution of the individual similarity values. Then, the weight \( w_1(D) \) of diagonal \( D \) is defined to be:

\[
w_1(D) = -\log P_1(l_D, S_D)
\]

As pointed out by Altschul and Erickson (1986), an important feature of this measure of segment similarity is that
the quality of segment pairs of different length can be compared. This is important for our approach, since we want to include diagonals of different length in our alignments.

However, from our point of view, this measure has a certain drawback: if a longer diagonal $D$ is split into several shorter diagonals $D_1, \ldots, D_n$, the sum of weights $\sum w_1(D_i)$ may equal or even exceed the weight $w_1(D)$. Thus, since the program tries to find collections of diagonals with maximum sum of weights, it would include $n$ diagonals $D_1, \ldots, D_n$ in the alignment instead of the single diagonal $D$. In DNA alignment, for example, the probability of a base pair representing a match is 0.25, so for a diagonal $D$ without mismatches, we would have $P_1(l_D, s_D) = 0.25^{l_D} \times \log 4$, i.e. the weight of such a diagonal would be proportional to its length. This means that a full-match diagonal of length $l$ would have the same weight as $l$ isolated single matches. Generally, DIALIGN 1 would compose alignments from many shorter diagonals rather than from fewer but longer ones. As a consequence, even significant local similarities may get lost in the ‘noise’ of small random diagonals.

To counterbalance this tendency, we modified the scoring scheme for DIALIGN 1 as follows. First, we introduced a user-defined threshold parameter $T$ such that a diagonal was considered for alignment only if its weight exceeded $T$. In addition, we required diagonals to have a minimum length of seven residues.

It is our experience that in situations where sequences are only locally related, DIALIGN 1 is clearly superior to conventional Needleman–Wunsch-based methods. If sequences are globally related, results of both methods are comparable (see Morgenstern et al., 1998a,b). Nevertheless, the dependency on the threshold $T$ was a serious drawback of the first version of the program since there was no general rule as to which value for $T$ would produce the best alignments.

To overcome these shortcomings, we have introduced a new weight function for diagonals (see Morgenstern et al., 1998b). Instead of considering the probability $P_1(l_D, s_D)$ of a given random diagonal having a sum of individual similarity values of at least $s_D$, we considered the probability $P_2(l_D, s_D)$ of finding any diagonal of length $l_D$ whose sum of individual similarity values is at least as large as $s_D$ somewhere within the comparison matrix of two random sequences of
1. iteration step

\[
\begin{array}{cccc}
D_1 & D_2 & D_3 & D_4 \\
\text{weight} & 0.2 & 2.6 & 4.7 & 2.2 \\
\text{overlap weight} & 0.2 & 5.3 & 4.7 & 4.9 \\
\end{array}
\]

2. iteration step

It is easy to see that using this new weight function greatly reduces the problems caused by ‘noisy’ random diagonals:

While the weight function \( w_1 \) gives a positive weight to almost every diagonal, the situation is different for the new weight function \( w_2 \): the probability of finding a diagonal with an average sum of similarity values somewhere in the comparison matrix is \( \approx 1 \), and therefore the weight of a random diagonal \( D_r \) is \( w_2(D_r) = -\ln 1 = 0 \). Since, even for closely related sequences, the vast majority of diagonals in a comparison matrix are random diagonals with no significant similarity among the two segments, replacing \( w_1 \) by \( w_2 \) not only mitigates the above-mentioned noise problem, but, by decreasing the number of diagonals under consideration, also reduces the computational costs of the program.

For high-scoring diagonals, we have derived the simple approximation formula \( w_2(D) = w_1(D) - K \), where \( K \) is a constant that depends on the sequence length (Morgenstern et al., 1998b). This prevents the program from splitting up long high-scoring diagonals into several smaller ones.
A new version of our program based on the weight function $w_2$ is called DIALIGN 2.

**Results and discussion**

If sequences are globally related, both versions of DIALIGN produce alignments comparable to those produced by the best standard methods such as CLUSTAL W (Thompson et al., 1994) and Divide and Conquer (Tönges et al., 1996; Stoye, 1998), as is shown in Morgenstern et al. (1998a,b).

Table 1 shows, however, that DIALIGN is superior to Needleman–Wunsch-based alignment methods if sequences are only locally related. This table also shows the main weakness of the original version of the program: here, it was necessary to use a positive threshold $T$ in order to obtain reasonable results. Without this threshold, i.e. with $T = 0$, local motifs would get lost in the ‘noise’ of small random diagonals. On the other hand, if $T$ is too large, even significant similarities are ignored. It seems that for globally related sequences, DIALIGN 1 produced best results without a threshold (Morgenstern et al., 1996, 1998a), whereas Table 1 indicates that, for locally related sequences, the best results were achieved with a threshold $T = 10$. By contrast, DIALIGN 2 produces satisfactory results without such a threshold parameter. Other alignment programs in this comparison are TWOALIGN (Abbeddaïm, 1997), ITERALIGN (Brocchieri and Karlin, 1998), CLUSTAL W (Thompson et al., 1994), DCA (Stoye, 1998), MULTALIN (Corpet, 1988), MAP (Huang, 1994), PIMA (Smith and Smith, 1990) and MATCH-BOX (Depiereux et al., 1997).

It should be mentioned that in Table 1 we have used only one single criterion to compare different alignment methods: we have compared their ability to find conserved motifs, i.e. their sensitivity. This is, in fact, the most important criterion if alignment methods are to be compared (see McClure et al., 1994). However, as pointed out by Briffeuil et al. (1998), another important aspect of alignment programs is their selectivity. A good alignment method should be able to align the related parts of sequences correctly, but also not to align their unrelated parts. The MATCH-BOX program, for example, has found only relatively few motifs in our test sequences. However, since this program aligned only small parts of the sequences, these results may be more valuable than the results of other methods where correctly aligned motifs are often hard to detect in large alignments of mostly unrelated sequences. The DIALIGN program, too, tries to align only the related parts of the sequences and to ignore the unrelated parts. We are planning to compare the selectivity of different alignment methods systematically in a forthcoming study.

### Table 1: Performance of different alignment methods applied to three different ‘local’ alignment problems. The table reports the ability to align correctly functional domains of the sequences. Entries in the table denote numbers of correctly aligned motifs. Multiple entries mean that a motif is correctly aligned within subgroups of sequences, but not between these subgroups. A motif is considered to be correctly aligned if at least 75% of the residues are correctly aligned. Test data sets are helix–turn–helix (Lawrence et al., 1993) acetyltransferases (Neuwald and Green, 1994) and basic helix–loop–helix transcription factors (database accession numbers P41894, Q02575, P17106, A55438, U10638, P13902, Q04635, U11444, A48085).

<table>
<thead>
<tr>
<th>Data set</th>
<th>HTH</th>
<th>Transferase</th>
<th>bHLH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>I</td>
<td>II</td>
<td>I</td>
</tr>
<tr>
<td>Number of sequences</td>
<td>30</td>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>Conserved domain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIALIGN 1, $T = 0$</td>
<td>6,6,3,2,2</td>
<td>12,2</td>
<td>9</td>
</tr>
<tr>
<td>DIALIGN 1, $T = 5$</td>
<td>7,7,3</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>DIALIGN 1, $T = 10$</td>
<td>19,2,2</td>
<td>16</td>
<td>13,2</td>
</tr>
<tr>
<td>DIALIGN 1, $T = 15$</td>
<td>16,5,2</td>
<td>12,2,2</td>
<td>6,2</td>
</tr>
<tr>
<td>DIALIGN 1, $T = 20$</td>
<td>5,4,3,2,2,2</td>
<td>5,3,2</td>
<td>3,2</td>
</tr>
<tr>
<td>DIALIGN 2, $T = 0$</td>
<td>24,2</td>
<td>16</td>
<td>14,2</td>
</tr>
<tr>
<td>DIALIGN 2, $T = 5$</td>
<td>18,3,2</td>
<td>14,2</td>
<td>10</td>
</tr>
<tr>
<td>TWOALIGN</td>
<td>10,6,3,3</td>
<td>13,2</td>
<td>6,5,2</td>
</tr>
<tr>
<td>ITERALIGN</td>
<td>16</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>CLUSTAL W</td>
<td>5,3,2,2,2</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>DCA</td>
<td>5,3,2,2,2,2</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>MULTALIN</td>
<td>6,5,4,2,2,2</td>
<td>8,3,2</td>
<td>7,5,2</td>
</tr>
<tr>
<td>MAP</td>
<td>6,5,4,3,2,2</td>
<td>7,2,2,2,2</td>
<td>4,4,2</td>
</tr>
<tr>
<td>PIMA</td>
<td>5,4,3,3,2,2</td>
<td>10,3,2</td>
<td>8,3,2</td>
</tr>
<tr>
<td>MATCH-BOX</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Two other newly developed programs performed remarkably well when applied to our test examples: TWOALIGN and ITERALIGN. Both methods are partly related to the DIALIGN approach. TWOALIGN employs a greedy approach in order to construct multiple alignments from local pairwise alignments. It uses the same data structures as DIALIGN to decide the question of consistency that arises in this context. ITERALIGN uses segment-to-segment comparison to find local pairwise alignments. However, segment pairs are not directly included in a multiple alignment, but they are used to construct ameliorated sequences in an iterative procedure. These sequences are then used to form blocks of segments for the final multiple alignment.

Table 2 compares the number and average length of diagonals in optimal pairwise alignments, and program running time on a HP 9000 machine (Model K460 2-CPU), for both versions of DIALIGN using three different data sets. The original version of our program, DIALIGN 1, tends to compose alignments from many short diagonals. Most diagonals are not much longer than the minimum length of seven residues, which indicates that the length of the selected diagonals is not determined by biological relatedness but by the, rather arbitrary, minimum length of seven residues, instead. In contrast, the new version, DIALIGN 2, selects fewer but longer diagonals. As follows from equation (1), the running time of the program depends on the number of diagonals in the respective optimal pairwise alignments. Since DIALIGN 2 composes alignments from fewer but longer diagonals, it is considerably faster than DIALIGN 1.

<table>
<thead>
<tr>
<th>Data set</th>
<th>N</th>
<th>L</th>
<th>Program version</th>
<th>$l_i$</th>
<th>$#.l_i$</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinase</td>
<td>6</td>
<td>281</td>
<td>DIALIGN 1</td>
<td>7.4</td>
<td>444</td>
<td>17.8s</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DIALIGN 2</td>
<td>25.5</td>
<td>97</td>
<td>8.6</td>
</tr>
<tr>
<td>Kinase</td>
<td>12</td>
<td>273</td>
<td>DIALIGN 1</td>
<td>7.5</td>
<td>1910</td>
<td>116</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DIALIGN 2</td>
<td>23.5</td>
<td>447</td>
<td>47</td>
</tr>
<tr>
<td>HTH</td>
<td>30</td>
<td>242</td>
<td>DIALIGN 1</td>
<td>7.4</td>
<td>8441</td>
<td>610</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>DIALIGN 2</td>
<td>21.1</td>
<td>1606</td>
<td>310</td>
</tr>
</tbody>
</table>

A time-consuming step in the program is updating the consistency bounds $b(x, i)$ and $b(x, i)$ whenever a new diagonal is accepted during the greedy procedure. The current version of DIALIGN uses a straightforward approach to calculate these values. Abdeddaim (1997) developed a more sophisticated solution for this problem that seems to be far more efficient than the direct approach we have used. We expect that applying his ideas to our segment-to-segment approach will further improve the running time of DIALIGN.

Finally, there is a general problem with our greedy strategy for multiple alignment: it may easily happen that the program runs into a local maximum. Once a ‘wrong’ diagonal is accepted for alignment, it cannot be removed later, though it may deteriorate the overall alignment score. Thus, it seems worthwhile to apply alternative techniques such as branch and cut (Reinet et al., 1997; Lenhof et al., 1999) or genetic algorithms (Notredame and Higgins, 1996) to the optimization problem defined by our segment-based objective function.

### Acknowledgements

I would like to thank Andreas Dress and William R. Atchley for many helpful discussions, Klaus Hahn for calculating the probability values $P_i$, Theresa Faus and Christopher Penkett for critically reading the manuscript, and three anonymous referees for their valuable comments. Dirk Evers, Folker Meyer, Alexander Sczyrba and Jens Stoye have created the WWW interface at the Bielefeld Bioinformatics Server (BiBiServ). Special thanks to Antje Krause and Knut Reinert for their help.

### References


