Speeding up the DIALIGN multiple alignment program by using the ‘Greedy Alignment of BIOlogical Sequences LIBrary’ (GABIOS-LIB)

Said Abdeddaïm$^1$ and Burkhard Morgenstern$^2$

1 LIFAR - ABISS, Faculté des Sciences et Techniques, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France,  
Said.Abdeddaïm@univ-rouen.fr  
2 AVENTIS Pharma, Rainham Road South, Essex RM10 7XS, UK.  
Present address: MIPS, Max-Planck-Institut für Biochemie, Am Klopferspitz 18a, 82152 Martinsried Germany  
morgenstern@mips.biochem.mpg.de

Abstract. A sensitive method for multiple sequence alignment should be able to align local motifs that are contained in some but not necessarily in all of the input sequences. In addition, it should be possible to integrate various of such partial local alignments into one single multiple output alignment. This leads to the question of consistency of partial alignments. Based on a new set-theoretical definition of sequence alignment, the consistency problem is discussed theoretically, and a recently developed library of C functions for consistency calculation (GABIOS-LIB) is described. GABIOS-LIB has been integrated into the DIALIGN alignment program to carry out consistency tests during the multiple alignment procedure. While the resulting alignments are exactly the same as with the previous version of DIALIGN, the running time of the program has been crucially improved. For large data sets, the new version of DIALIGN is up to 120 times faster than the old version.  
Keywords: Multiple Sequence Alignment, Partial Alignment, Consistency, Consistent Equivalence Relation, Greedy Algorithm.

1 Introduction

Traditionally, there are two different approaches to sequence alignment: global methods that align sequences over their entire length [8,21,26,25] and local methods that try to align the most highly conserved sub-regions of the input sequences [24,23,3,13]. One problem with these approaches is that it is often not known in advance if sequences are globally or only locally related. A versatile alignment tool should align those regions of the input sequences that are sufficiently similar to each other but it would not try to align the non-related parts of the sequences. Thus, such a program would return a global alignment whenever sequences are globally related but a local alignment if only local homology can be
detected. One possible way to achieve this is to integrate statistically significant (partial) local alignments $P_1, \ldots, P_k$ into one resulting output alignment $A$.

The idea to generate sequence alignments by combining partial alignments of local similarities is not new. Various authors have proposed to generate pairwise local or global alignments by chaining fragment alignments, see Wilbur and Lipman [32], Eppstein et al. [7], and Chao and Miller [4]. These authors have developed time and space efficient fragment-chaining algorithms for near-optimal alignment in the sense of the traditional Needleman-Wunsch [21] and Smith-Waterman [24] objective functions. Joseph et al. [11] have proposed a greedy algorithm that is based on statistically significant segment pairs.

Algorithms that integrate local alignments have also been proposed for multiple alignment. Here, the problem is to decide whether a collection of local alignments is consistent. Informally, we call a set $\{P_1, \ldots, P_k\}$ of partial alignments consistent if an alignment $A$ of the input sequences exists such that every pair of residues that is aligned by one or several of the alignments $P_i$ is also aligned by $A$. A formal definition of our notion of consistency will be given in the next section. The question of consistency is easy to decide if each local alignment $P_i$ involves all of the input sequences. Vingron and Argos [30], Vingron and Pevzner [31] and Depierreux et al. [6, 5] have proposed multiple alignment methods that search for motifs that are simultaneously contained in all input sequences. In this case, a sufficient condition for consistency is that, for any two local alignments $P_i$ and $P_j$, either $P_i \prec P_j$ or $P_j \prec P_i$ holds where $P_i \prec P_j$ means that in every sequence, residues aligned by $P_i$ are to the left of residues aligned by $P_j$. From a biological point of view, however, it is desirable to allow for homologies involving not all but only some of the input sequences. A multiple alignment program that finds only those similarities that are present in all sequences in a given input data set will necessarily miss many biologically important homologies.

Recently, we have introduced three heuristics for multiple alignment that integrate partial local alignments not necessarily involving all of the input sequences. These methods generate multiple alignments in a greedy way by incorporating local partial alignments one-by-one into a resulting multiple alignment. SONDALIGN [1] assembles multiple alignments from blocks of un-gapped local alignments that may involve two or more sequences, DIALIGN [15, 17, 18] uses un-gapped segment pairs - so-called fragments or diagonals –, and TWOALIGN [2] combines pairwise local alignments in the sense of Smith and Waterman [24] to obtain a final multiple alignment. During the greedy procedures, these three programs have to reject new partial alignments for consistency with those alignments that have already been accepted for the final alignment. To this end, they store and update certain data structures that are called transitivity frontiers for SONDALIGN and TWOALIGN and consistency bounds for DIALIGN.

DIALIGN has been successfully used to detect local homologies in nucleic acid and protein sequences. In a recent study, Götgens et al. [10] have used DIALIGN to align large genomic sequences from human, mouse and chicken. In the human/mouse alignment, multiple peaks of homology were found, some
of which precisely correspond to known enhancers. In addition, the DIALIGN multi-alignment of human, mouse and chicken revealed a new region of homology that was then experimentally shown to be a previously unknown enhancer, see also Miller [14] for a discussion of these results. Thompson et al. [28], have used the BALIBASE of benchmark alignments [27] to systematically compare the most widely used programs for multiple protein alignment. Here, DIALIGN was reported to be the most successful local method. It also performed well on globally related sequence sets though here CLUSTAL W [26], PRRP [9] and SAGA [22] were superior. The paper by Thompson et al., however, addressed also a major weakness of DIALIGN: it is considerably slower than progressive alignment methods. Aligning a set of 89 histone sequences took as much as 13,649 s with DIALIGN compared to 161 s with CLUSTAL. This may not be a serious problem if only a single protein family is studied. However, with the huge amount of genomic sequence data that are now available, automatic alignment of whole databases of sequence families has become a routine task, see, for example, [12, 29]. Here, program running time is a crucial criterion in choosing a suitable alignment method.

Test runs have shown that for large sequence sets, the procedure of updating the consistency bounds was by far the most time-consuming step in previous versions of DIALIGN. Abdelhám has recently developed a library of C functions called GABIOS-LIB (Greedy Alignment of BIOlogical Sequences Library) that can be used to efficiently calculate the transitivity frontiers and to consistency-check (partial) local alignments that may have been produced by arbitrary methods. We have integrated GABIOS-LIB into DIALIGN to speed up the consistency check for fragments (segment pairs). In the present paper, the time and space complexity of GABIOS-LIB is analysed theoretically and compared to the method that was previously used in DIALIGN. Experiments with both artificial and real sequence data demonstrate that GABIOS-LIB is far more efficient than the previous procedure. In our test examples, the new version 2.1 of DIALIGN is up to 120 times faster than version 2.0 while the resulting alignments are exactly the same. In addition, GABIOS-LIB has reduced the amount of computer memory used by DIALIGN.

2 The consistency problem for partial alignments

2.1 Definitions and notations

Let $S = \{S_1, \ldots, S_N\}$ be a sequence family and let $X$ be the set of all sites of $S$ where a site $x = [i, p]$ represents the $p$-th position in the $i$-th sequence. On $X$, we define a partial order relation $\preceq$ such that for any two sites $x = [i, p]$ and $x' = [i', p']$, $x \preceq x'$ holds if and only if both $i = i'$ and $p \leq p'$ are true. In the language of order theory, $\preceq$ is the direct sum of the ‘natural’ linear order relations that are given on the individual sequences. Every binary relation $R$ on $X$ extends the relation $\preceq$ to a quasi order relation $\preceq_R = (\preceq \cup R)_t$ on $X$, where $S_t$ denotes the transitive closure of a relation $S$, i.e. the smallest transitive relation containing $S$, see Figure 1.
Fig. 1. A relation $R = \{(v, w), (x, y)\}$ (represented by arrows) defined on the set $X$ of all sites (black dots) of a sequence family $S = \{S_1, S_2, S_3\}$ extends the partial order relation $\preceq$ on $X$ to a quasi partial ordering $\preceq_R = (\preceq \cup R)_R$. We have, for example, $u \preceq v$, $vRw$, $w \preceq x$, $xRy$, $y \preceq z$ and therefore $u \preceq_R z$.

We call a relation $R$ on $X$ consistent if the extended relation $\preceq_R$ preserves the linear order relations on the individual sequences, formally: if all restrictions of $\preceq_R$ to the individual sequences coincide with their respective 'natural' linear order relations. In other words, the requirement is that for any two sites $x$ and $y$ that belong to the same sequence, $x \preceq_R y$ implies $x \preceq y$. Moreover, we call a set $\{R_1, \ldots, R_n\}$ of relations consistent if the union $\bigcup_i R_i$ is consistent, we say that $R_1$ is consistent with $R_2$ if $\{R_1, R_2\}$ is consistent, and a pair $(x, y) \in X^2$ is called consistent with a relation $R$ if $\{(x, y)\}$ is consistent with $R$.

As proposed in [17], a (partial) alignment $A$ of the family $S$ can be defined as a consistent equivalence relation on the set $X$ where we write $(x, y) \in A$ or $xAy$ if the sites $x$ and $y$ are either aligned by $A$ or identical. As an equivalence relation, an alignment $A$ partitions $X$ into equivalence classes $[x]_A = \{y \in X : (x, y) \in A\}$. It can be shown that an equivalence relation $A$ on $X$ is an alignment in the sense of the above definition if and only if it is possible to introduce gap characters into the sequences $S_i$ such that the equivalence classes $[x]_A, x \in X$ are precisely those sets of sites that are in the same column of the resulting two-dimensional array, see [20] for more details.

A common feature of greedy multiple alignment algorithms is that they include partial alignments $P_1, \ldots, P_k$ one after the other into a growing multiple alignment – always provided that a new alignment $P_i$ is consistent with those alignments that have been included previously. Formally, a monotonously increasing set $A_1 \subset \ldots \subset A_k$ of alignments is defined by

$$
A_1 = P_1
$$

$$
A_i = \begin{cases} 
(A_{i-1} \cup P_i)_{e} & \text{if } P_i \text{ is consistent with } A_{i-1} \\
A_{i-1} & \text{otherwise}, 
\end{cases} \quad i = 2, \ldots, k. \quad (1)
$$

A final alignment $A$ is then obtained as the largest alignment $A = A_k$ of this set. Therefore, every greedy alignment approach has to resolve the question of consistency: at any stage of the alignment procedure, it must be known which pairs
$(x, y) \in X^2$ are still alignable without leading to inconsistencies with the current alignment $A_i$, i.e. with those pairs of sites that have already been accepted for the final alignment.

2.2 Transitivity frontiers and consistency bounds

An alignment $A$ of a sequence family $S$ imposes for every site $x \in X$ and every sequence $S_i \in S$ a lower bound $\underline{b}_A(x, i)$ and an upper bound $\overline{b}_A(x, i)$ such that a site $y = [i, p] \in S_i$ is alignable with $x$ without leading to inconsistencies with $A$ if and only if $\underline{b}_A(x, i) \leq p \leq \overline{b}_A(x, i)$ holds, see Figure 2 for an example. Formally, we define

$$\underline{b}_A(x, i) = \min \{ p : (x, [i, p]) \text{ consistent with } A \}$$

and

$$\overline{b}_A(x, i) = \max \{ p : (x, [i, p]) \text{ consistent with } A \}.$$

In order to test fragments for consistency during the greedy procedure, previous versions of DIALIGN calculated and updated these consistency bounds.

![Diagram](image-url)

Fig. 2. For an alignment $A$ (bold lines) and a site $x$, the transitivity frontiers with respect to a sequence $S_i$ coincide with the corresponding consistency bounds if $x$ is aligned to some site in $S_i$. For example, site $u$ in $S_2$ is aligned with $x$, so we have $\overline{b}_A(x, 2) = \text{Succ}_A(x, 2) = 6$. In sequence $S_1$, on the other hand, $v$ is the right-most site that can be aligned with $x$, but $w$ is the left-most site with $x \preceq_A w$, so we have $\overline{b}_A(x, 1) = 7$ but $\text{Succ}_A(x, 1) = 8$.

GABIOS-LIB is using a so-called transitivity frontiers to carry out the same consistency check. Here, the predecessor frontier $\text{Pred}_A(x, i)$ is defined as the position of the right-most site $y$ in sequence $S_i$ such that $y \preceq_A x$ is true, and the successor frontier $\text{Succ}_A(x, i)$ is defined accordingly as the position of the
VI

left-most site \( y \) in sequence \( S_i \) with \( x \leq_A y \) so we have

\[
\text{Pred}_A(x, i) = \max\{p : [i, p] \leq_A x\}
\]

and

\[
\text{Succ}_A(x, i) = \min\{p : x \leq_A [i, p]\}.
\]

The transitivity frontiers are related to the consistency bounds in the following way: if \( x \) is already aligned to some site \([i, p]\) in sequence \( S_i \), then the predecessor and successor frontiers of \( x \) with respect to \( S_i \) both equal \( p \) and they \textit{coincide} with the consistency bounds, i.e. one has

\[
\text{Pred}_A(x, i) = \text{Succ}_A(x, i) = \underline{b}_A(x, i) = \overline{b}_A(x, i) = p.
\]

This is, for example, the case for the frontiers and bounds of \( x \) with respect to \( S_2 \) in Figure 2. In contrast, if no site in \( S_i \) is aligned with \( x \), one has

\[
\text{Pred}_A(x, i) = \underline{b}_A(x, i) - 1
\]

and

\[
\text{Succ}_A(x, i) = \overline{b}_A(x, i) + 1
\]

as is the case with the corresponding frontiers and bounds with respect to \( S_1 \) in Figure 2. Therefore, if it is known for every site \( x \) and every sequence \( S_i \) whether \( x \) is aligned with some site in \( S_i \), transitivity frontiers are easily obtained from the consistency bounds and vice versa, so both data structures are equivalent in that they contain the same information about which residue pairs are alignable under the consistency constraints imposed by a given alignment \( A \).

3 GABIOS-LIB

The Greedy Alignment of BIOlogical Sequences LIbrary (GABIOS-LIB) is a set of functions implemented in ANSI C by Abeddaim. These functions can be used by any greedy alignment program in order to test in constant time which sites in a sequence \( S_j \) are alignable with a site \( x \) of another sequence \( S_i \). Each time two sites are aligned during the greedy procedure, GABIOS-LIB updates the transitivity frontiers using the incremental algorithm Edge-Addition presented in [2]. In addition, GABIOS-LIB uses some ideas first introduced in [1] to further reduce computing time and memory.

In this section, we discuss how the successor frontiers \( \text{Succ}_A(x, i) \) for an alignment \( A \) are affected if a (partial) alignment \( P \) is added to \( A \) and how the frontiers \( \text{Succ}_B(x, i) \) for the resulting alignment \( B = (A \cup P)_e \) can be calculated. For symmetry reasons, all results apply to the predecessor frontiers as well.
3.1 The incremental algorithm EdgeAddition

First of all, a simple but important observation is that if a new partial alignment \( P \) is added to an existing alignment \( A \), the frontiers with respect to the new alignment \( B = (A \cup P)_e \) need to be calculated only if \( B \) is actually different from \( A \) which is the case if and only if \( P \) is not already a subset of \( A \). EdgeAddition stores for every site \( x \) and every sequence \( S_j \) the information whether \( x \) is already aligned with some residue from sequence \( S_j \); this information is used to check if a new alignment \( P \) is already contained in \( A \).

If and only if \( P \not\subseteq A \) holds, the transitivity frontiers \( Succ_B \) are different from the frontiers \( Succ_A \). In general, however, the frontiers will change not for all but only for some sites, and the computing time can be minimized by identifying those sites. For simplicity, we consider the simplest case where a single pair of sites \((x, y)\) is added to \( A \).

**Observation 1** Let \( A \) be an alignment of a sequence family \( S \) and \((x, y)\) a pair of sites that is consistent with \( A \). Let \( B = (A \cup \{(x, y)\})_e \) be the alignment that is obtained by adding \((x, y)\) to \( A \). Then for every site \( u \) in a sequence \( S_i \) and for every sequence \( S_j \) the successor frontiers of \( B \) are

\[
Succ_B(u, j) = \begin{cases} 
\min\{Succ_A(u, j), Succ_A(y, j)\} & \text{if } u \not\preceq_A x \\
\min\{Succ_A(u, j), Succ_A(x, j)\} & \text{if } u \not\preceq_A y \\
Succ_A(u, j) & \text{otherwise.}
\end{cases}
\]

It follows that the transitivity frontiers can change only for those sites \( u \) for which either \( u \not\preceq_A x \) or \( u \not\preceq_A y \) is true.

3.2 Further reduction of computing time and memory

In order to further reduce the computational costs for calculating the consistency frontiers, GABIOS-LIB uses the following two facts. (1) If two sites \( x \) and \( y \) are aligned by \( A \), i.e., if \( xAy \) holds, then they have necessarily the same frontiers \( Succ_A(x, i) = Succ_A(y, i) \) for \( i = 1, \ldots, N \). Therefore, rather than processing the transitivity frontiers for all individual sites \( x \), GABIOS-LIB stores and updates the frontiers for those equivalence classes \([x]_A\) that consist of more than one single site. (2) Let \( x = [i, p] \) be an orphan site in the \( i \)-th sequence, i.e., a site that is not aligned with any other site \( y \not= x \). Then the successor frontiers \( Succ_A(x, j) \) with respect to all sequences \( S_j \not= S_i \), coincide with the corresponding frontiers of the left-most non-orphan site \( y = [i, p'] \) with \( p' > p \). In Figure 2, for example, \( v = [1, 7] \) is an orphan site in \( S_1 \). The left-most non-orphan site \([1, p']\) with \( p' > 7 \), is the site \( w = [1, 8] \). Therefore for \( j \not= 1 \), the successor frontiers \( Succ_A(v, j) \) coincide with the corresponding frontiers for \( w \), i.e., we have \( Succ_A(v, j) = Succ_A(w, j) \) for \( j = 2, 3, 4 \). Thus, instead of storing the transitivity frontiers for an orphan site \( x \), the corresponding site \( y \) can be stored in a tabular nextClass by defining \( nextClass[x] = p' \). This way, the frontiers \( Succ_A(x, j) \) of an orphan site \( x \) can be established in constant time each time a new pair of sites is aligned.
4 Time and space efficient multiple segment-by-segment alignment

<table>
<thead>
<tr>
<th></th>
<th>DIALIGN 2.0</th>
<th></th>
<th>DIALIGN 2.1</th>
<th></th>
<th>2.0/2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$t_0$</td>
<td>$t_0/N^3$</td>
<td>$t_0/N^4$</td>
<td>$t_1$</td>
<td>$t_1/N^2$</td>
</tr>
<tr>
<td>25</td>
<td>22</td>
<td>1.4 $10^{-3}$</td>
<td>5.6 $10^{-5}$</td>
<td>10</td>
<td>0.016</td>
</tr>
<tr>
<td>50</td>
<td>168</td>
<td>1.3 $10^{-3}$</td>
<td>2.6 $10^{-5}$</td>
<td>41</td>
<td>0.016</td>
</tr>
<tr>
<td>(I)</td>
<td>75</td>
<td>601</td>
<td>1.4 $10^{-3}$</td>
<td>1.8 $10^{-5}$</td>
<td>100</td>
</tr>
<tr>
<td>100</td>
<td>1402</td>
<td>1.4 $10^{-3}$</td>
<td>1.4 $10^{-5}$</td>
<td>220</td>
<td>0.022</td>
</tr>
<tr>
<td>150</td>
<td>5725</td>
<td>1.6 $10^{-3}$</td>
<td>1.1 $10^{-5}$</td>
<td>576</td>
<td>0.025</td>
</tr>
<tr>
<td>200</td>
<td>14353</td>
<td>1.7 $10^{-3}$</td>
<td>8.9 $10^{-6}$</td>
<td>1874</td>
<td>0.046</td>
</tr>
<tr>
<td>(II)</td>
<td>25</td>
<td>46</td>
<td>2.9 $10^{-3}$</td>
<td>1.1 $10^{-4}$</td>
<td>10</td>
</tr>
<tr>
<td>50</td>
<td>640</td>
<td>5.1 $10^{-3}$</td>
<td>1.0 $10^{-4}$</td>
<td>43</td>
<td>0.017</td>
</tr>
<tr>
<td>100</td>
<td>10557</td>
<td>1.0 $10^{-2}$</td>
<td>1.0 $10^{-4}$</td>
<td>200</td>
<td>0.020</td>
</tr>
<tr>
<td>150</td>
<td>52431</td>
<td>1.5 $10^{-2}$</td>
<td>1.0 $10^{-4}$</td>
<td>555</td>
<td>0.024</td>
</tr>
<tr>
<td>200</td>
<td>177423</td>
<td>2.2 $10^{-2}$</td>
<td>1.1 $10^{-4}$</td>
<td>1429</td>
<td>0.035</td>
</tr>
<tr>
<td>(III)</td>
<td>25</td>
<td>25</td>
<td>1.6 $10^{-3}$</td>
<td>6.4 $10^{-5}$</td>
<td>7</td>
</tr>
<tr>
<td>50</td>
<td>341</td>
<td>2.7 $10^{-3}$</td>
<td>5.4 $10^{-5}$</td>
<td>29</td>
<td>0.011</td>
</tr>
<tr>
<td>75</td>
<td>1507</td>
<td>3.7 $10^{-3}$</td>
<td>5.0 $10^{-5}$</td>
<td>73</td>
<td>0.012</td>
</tr>
<tr>
<td>100</td>
<td>5046</td>
<td>5.0 $10^{-3}$</td>
<td>5.0 $10^{-5}$</td>
<td>147</td>
<td>0.014</td>
</tr>
<tr>
<td>(IV)</td>
<td>25</td>
<td>61</td>
<td>3.9 $10^{-3}$</td>
<td>1.5 $10^{-4}$</td>
<td>14</td>
</tr>
<tr>
<td>50</td>
<td>844</td>
<td>6.7 $10^{-3}$</td>
<td>1.3 $10^{-4}$</td>
<td>63</td>
<td>0.025</td>
</tr>
<tr>
<td>75</td>
<td>4157</td>
<td>9.8 $10^{-3}$</td>
<td>1.3 $10^{-4}$</td>
<td>149</td>
<td>0.026</td>
</tr>
<tr>
<td>100</td>
<td>11619</td>
<td>1.1 $10^{-2}$</td>
<td>1.1 $10^{-4}$</td>
<td>288</td>
<td>0.028</td>
</tr>
</tbody>
</table>

Table 1. Running time $t_0$ and $t_1$ of versions 2.0 and 2.1 of DIALIGN. Version 2.0 is using the old method of calculating the consistency bounds while version 2.1 is calculating the transitivity frontiers using GABIOS-LIB. Sequences are (I) independent and (II) identical random sequences of length 100, (III) Immunoglobulin domain sequences of average length 63.3 and 20% average identity, and (IV) Ribosomal protein L7/L12 C-terminal domain sequences of average length 119.3 and 53% average identity. The running time improvement achieved by GABIOS-LIB strongly increases with the number $N$ of sequences to be aligned. For the two sets of real-world sequences, figures are comparable to the case of identical random sequences where an improvement of up to 120 was achieved.

In order to construct a multiple alignment of a sequence family ${\mathcal S}$, DIALIGN calculates in a first step optimal pairwise alignments for all possible pairs of input sequences as explained in [16]. Optimality, in this context, refers to the segment-based objective function used in DIALIGN as defined in [19,15], i.e. an optimal pairwise alignment is a chain of fragments (gap-free segment pairs) $f_1 \ll \ldots \ll f_k$ such that $\sum_{i=1}^{k} w(f_i)$ is maximal. Here, $w$ is a weighting function.
defined on the set of all possible fragments, and \( f_i \ll f_j \) means that the end positions of \( f_i \) are both strictly smaller than the respective beginning positions of \( f_j \). Fragments from the respective optimal pairwise alignments are then greedily integrated into a resulting multiple alignment \( A \). Let \( L \) be the maximum length of the sequences \( S_1, \ldots, S_N \). Since \( O(N^2) \) pairwise alignments are performed each of which consisting of at most \( L \) fragments, \( O(N^2 L) \) fragments are to be checked for consistency. Updating the consistency bounds takes \( O(N^2) \) time if a new fragment is accepted for alignment. Since previous versions of DIALIGN calculated the consistency bounds for every fragment that was accepted for alignment, the worst-case time complexity of the entire greedy procedure was \( O(N^4 L) \). Table 1 shows that the running time of DIALIGN 2.0 is in fact proportional to \( N^4 \) if identical sequences are aligned where all fragments from the optimal pairwise alignments are necessarily consistent and are therefore integrated into \( A \).

Let us consider a consistent set of pairs \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \) that are successively integrated into a growing set of alignments \( A_1 \subset \ldots \subset A_m \) by defining \( A_i = \{(x_1, y_1), \ldots, (x_i, y_i)\} \). It was shown in [2] that if each pair \( (x_i, y_i) \) is actually new, i.e. not yet contained in the previous alignment \( A_{i-1} \), then GABIOS-LIB takes \( O(N^2 m + |X|^2) \) time to update the transitivity frontiers during the procedure of integrating all \( m \) pairs. It was also explained in [2] that there can be at most \( |X| \) new pairs of sites – every additional pair would either be inconsistent or would already be contained in the current alignment. Therefore, GABIOS-LIB takes \( O(N^2 |X| + |X|^2) = O(N^3 L + N^2 L^2) \) time to compute the transitivity frontiers while integrating an arbitrary set of pairs. Note that this complexity analysis is a worst-case estimate. As shown in Table 1, the real running time of GABIOS-LIB is better than \( O(N^3) \). The new version 2.1 of DIALIGN is still slower than the most widely used multi-alignment program CLUSTAL W [26], but the difference in running time is now reduced to a factor of about 10. Parameter optimization should further decrease the running time of DIALIGN.

<table>
<thead>
<tr>
<th>( N )</th>
<th>#CB</th>
<th>#TF</th>
<th>#CB #TF</th>
<th>#CB</th>
<th>#TF</th>
<th>#CB #TF</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>125,000</td>
<td>5,000</td>
<td>25.0</td>
<td>85,700</td>
<td>11,500</td>
<td>7.5</td>
</tr>
<tr>
<td>50</td>
<td>(I) 500,000</td>
<td>10,000</td>
<td>50.0 (III)</td>
<td>346,600</td>
<td>43,900</td>
<td>7.9</td>
</tr>
<tr>
<td>75</td>
<td>1,125,000</td>
<td>15,000</td>
<td>75.0</td>
<td>780,000</td>
<td>76,650</td>
<td>10.2</td>
</tr>
<tr>
<td>100</td>
<td>2,000,000</td>
<td>20,000</td>
<td>100.0</td>
<td>1,382,800</td>
<td>142,400</td>
<td>9.7</td>
</tr>
<tr>
<td>25</td>
<td>125,000</td>
<td>28,050</td>
<td>4.5</td>
<td>148,800</td>
<td>13,300</td>
<td>11.2</td>
</tr>
<tr>
<td>50</td>
<td>(II) 500,000</td>
<td>106,800</td>
<td>4.7 (IV)</td>
<td>609,600</td>
<td>75,600</td>
<td>8.1</td>
</tr>
<tr>
<td>75</td>
<td>1,125,000</td>
<td>204,150</td>
<td>5.5</td>
<td>1,356,450</td>
<td>145,350</td>
<td>9.3</td>
</tr>
<tr>
<td>100</td>
<td>2,000,000</td>
<td>375,000</td>
<td>5.3</td>
<td>2,403,800</td>
<td>212,200</td>
<td>11.3</td>
</tr>
</tbody>
</table>

Table 2. Number of integers allocated by DIALIGN 2.0 and DIALIGN 2.1 for storing consistency bounds (\#CB) and transitivity frontiers (\#TF), respectively. Sequence sets are as in Table 1.
In the previous version of DIALIGN, the consistency bounds were stored for every site \(x \in X\). Since for every \(x\), \(2N\) integer values had to be considered – upper and lower bounds with respect to all \(N\) sequences –, computer memory had to be allocated for exactly \(2N|X|\) integer values. By contrast, GABIOS-LIB stores \(2N\) transitivity frontiers only for non-orphan equivalence classes. For orphan sites, single integers are stored that refer to the next non-orphan site in the same sequence. In the worst case, the number of non-orphan equivalence classes is in the order of \(|X|\), so the space complexity for GABIOS-LIB is upper-bounded by \(O(|X|N) = O(N^2L)\) which was also the real space complexity of the old version of DIALIGN. Table 2 shows, however, that the actual memory requirement for storing the transitivity frontiers with GABIOS-LIB is far smaller than for storing the consistency bounds in DIALIGN 2.0.

References


