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

Saïd Abdeddaïm[1] and Burkhard Morgenstern[2]

[1] LIFAR - ABISS, Faculté des Sciences et Techniques, Université de Rouen, 76821 Mont-Saint-Aignan Cedex, France,
Said.Abdeddaim@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.
Availability: http://bibiserv.TechFak.Uni-Bielefeld.DE/dialign/
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 Depiereux *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 $P_i \ll P_j$ or $P_j \ll P_i$ holds where $P_i \ll P_j$ 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. SOUNDALIGN [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 test 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 SOUNDALIGN 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, Göttgens *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 data bases 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. Abdeddaïm 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 $\mathcal{S} = \{S_1, \ldots, S_N\}$ be a sequence family and let X be the set of all *sites* of $\mathcal{S}$ 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.

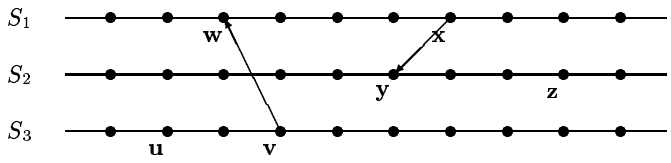

Fig. 1. A relation $R = \{(v, w), (x, y)\}$ (represented by arrows) defined on the set X of all *sites* (black dots) of a sequence family $\mathcal{S} = \{S_1, S_2, S_3\}$ *extends* the *partial* order relation $\preceq$ on X to a *quasi partial* ordering $\preceq_R = (\preceq \cup R)_t$. We have, for 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 $\cup_i R_i$ 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 $\mathcal{S}$ 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

$$A_1 = P_1$$
$$A_i = \begin{cases} (A_{i-1} \cup P_i)_e & \text{if } P_i \text{ is consistent with } A_{i-1} \\ A_{i-1} & \text{otherwise,} \end{cases} \quad i = 2, \ldots, k. \qquad (1)$$

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 $\mathcal{S}$ imposes for every site $x \in X$ and every sequence $S_i \in \mathcal{S}$ 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*.

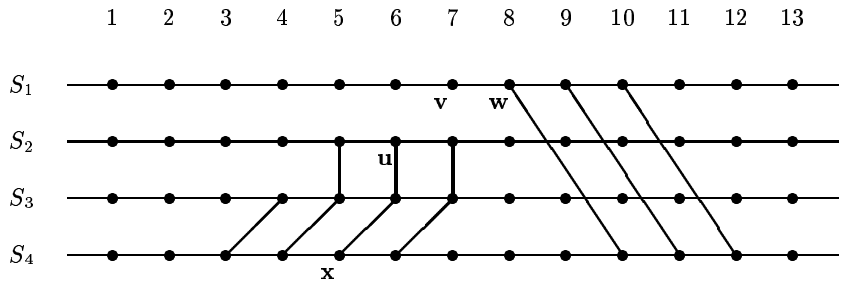

Fig. 2. For an alignment A (bold lines) and a site x, the transitivity frontiers with respect to a sequence S_i *coincide* with the corresponding consistency bounds if x is aligned to some site in S_i. For example, site u in S_2 is aligned with x, so we have $\overline{b}_A(x, 2) = Succ_A(x, 2) = 6$. 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 $Succ_A(x, 1) = 8$.

GABIOS-LIB is using a so-called *transitivity frontiers* to carry out the same consistency check. Here, the *predecessor frontier* $Pred_A(x, i)$ 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* $Succ_A(x, i)$ is defined accordingly as the position of the

left-most site y in sequence S_i with $x \preceq_A y$ so we have

$$Pred_A(x,i) = \max\{p : [i,p] \preceq_A x\}$$

and

$$Succ_A(x,i) = \min\{p : x \preceq_A [i,p]\}.$$

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 *coincide* with the consistency bounds, i.e. one has

$$Pred_A(x,i) = Succ_A(x,i) = \underline{b}_A(x,i) = \overline{b}_A(x,i) = p.$$

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

$$Pred_A(x,i) = \underline{b}_A(x,i) - 1$$

and

$$Succ_A(x,i) = \overline{b}_A(x,i) + 1$$

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 Abdeddaïm. 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 an other sequence S_i. Each time two sites are aligned during the greedy procedure, GABIOS-LIB updates the transitivity frontiers using the incremental algorithm EdgeAddition 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 $Succ_A(x,i)$ for an alignment A are affected if a (partial) alignment P is added to A and how the frontiers $Succ_B(x,i)$ 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 $P \not\subset A$ 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 $\mathcal{S}$ 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*

$$Succ_B(u, j) = \begin{cases} \min\{Succ_A(u, j), Succ_A(y, j)\} & \text{if } u \preceq_A x \\ \min\{Succ_A(u, j), Succ_A(x, j)\} & \text{if } u \preceq_A y \\ Succ_A(u, j) & \text{otherwise.} \end{cases}$$

It follows that the transitivity frontiers can change only for those sites u for which either $u \preceq_A x$ or $u \preceq_A y$ 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 $y \neq x$. Then the successor frontiers $Succ_A(x, j)$ with respect to all sequences $S_j \neq S_i$, *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 $j \neq 1$, 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

		DIALIGN 2.0			DIALIGN 2.1			2.0/2.1
	N	t_0	t_0/N^3	t_0/N^4	t_1	t_1/N^2	t_1/N^3	t_0/t_1
	25	22	$1.4\ 10^{-3}$	$5.6\ 10^{-5}$	10	0.016	$6.4\ 10^{-4}$	2.2
	50	168	$1.3\ 10^{-3}$	$2.6\ 10^{-5}$	41	0.016	$3.3\ 10^{-4}$	4.1
(I)	75	601	$1.4\ 10^{-3}$	$1.8\ 10^{-5}$	100	0.017	$2.3\ 10^{-4}$	6.0
	100	1402	$1.4\ 10^{-3}$	$1.4\ 10^{-5}$	220	0.022	$2.2\ 10^{-4}$	6.8
	150	5725	$1.6\ 10^{-3}$	$1.1\ 10^{-5}$	576	0.025	$1.7\ 10^{-4}$	10.0
	200	14353	$1.7\ 10^{-3}$	$8.9\ 10^{-6}$	1874	0.046	$2.3\ 10^{-4}$	7.6
	25	46	$2.9\ 10^{-3}$	$1.1\ 10^{-4}$	10	0.016	$6.5\ 10^{-4}$	4.6
	50	640	$5.1\ 10^{-3}$	$1.0\ 10^{-4}$	43	0.017	$3.4\ 10^{-4}$	14.9
(II)	75	3282	$7.7\ 10^{-3}$	$1.0\ 10^{-4}$	104	0.018	$2.4\ 10^{-4}$	31.6
	100	10557	$1.0\ 10^{-2}$	$1.0\ 10^{-4}$	200	0.020	$2.0\ 10^{-4}$	52.8
	150	52431	$1.5\ 10^{-2}$	$1.0\ 10^{-4}$	555	0.024	$1.6\ 10^{-4}$	94.5
	200	177423	$2.2\ 10^{-2}$	$1.1\ 10^{-4}$	1429	0.035	$1.7\ 10^{-4}$	124.2
	25	25	$1.6\ 10^{-3}$	$6.4\ 10^{-5}$	7	0.011	$4.4\ 10^{-4}$	3.5
(III)	50	341	$2.7\ 10^{-3}$	$5.4\ 10^{-5}$	29	0.011	$2.3\ 10^{-4}$	11.7
	75	1597	$3.7\ 10^{-3}$	$5.0\ 10^{-5}$	73	0.012	$1.7\ 10^{-4}$	21.8
	100	5046	$5.0\ 10^{-3}$	$5.0\ 10^{-5}$	147	0.014	$1.4\ 10^{-4}$	34.3
	25	61	$3.9\ 10^{-3}$	$1.5\ 10^{-4}$	14	0.022	$8.9\ 10^{-4}$	4.3
(IV)	50	844	$6.7\ 10^{-3}$	$1.3\ 10^{-4}$	63	0.025	$5.0\ 10^{-4}$	13.3
	75	4157	$9.8\ 10^{-3}$	$1.3\ 10^{-4}$	149	0.026	$3.5\ 10^{-4}$	27.8
	100	11619	$1.1\ 10^{-2}$	$1.1\ 10^{-4}$	288	0.028	$2.8\ 10^{-4}$	40.3

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 $A_i = \{(x_1, y_1), \ldots, (x_i, y_i)\}_e$. 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.

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

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 DIALING 2.0.

References

1. S. Abdeddaïm. Fast and sound two-step algorithms for multiple alignment of nucleic sequences. In *Proceedings of the IEEE International Joint Symposia on Intelligence and Systems*, pages 4–11, 1996.
2. S. Abdeddaïm. Incremental computation of transitive closure and greedy alignment. In *Proc. of 8-th Annual Symposium on Combinatorial Pattern Matching*, volume 1264 of *Lecture Notes in Computer Science*, pages 167–179, Heidelberg, 1997. Springer Verlag.
3. S. F. Altschul, W. Gish, W. Miller, E. M. Myers, and D. J. Lipman. Basic local alignment search tool. *J. Mol. Biol.*, 215:403–410, 1990.
4. K.-M. Chao and W. Miller. Linear-space algorithms that build local alignments from fragments. *Algorithmica*, 13:106–134, 1995.
5. E. Depiereux, G. Baudoux, P. Briffeuil, I. Reginster, X. D. Boll, C. Vinals, and E. Feytmans. Match-Box server: a multiple sequence alignment tool placing emphasis on reliability. *CABIOS*, 13:249–256, 1997.
6. E. Depiereux and E. Feytmans. Match-box: a fundamentally new algorithm for the simultaneous alignment of several protein sequences. *CABIOS*, 8:501–509, 1992.
7. D. Eppstein, Z. Galil, R. Giancarlo, and G. Italiano. Sparse dynamic programming I: Linear cost functions. *J. Assoc. Comput. Mach.*, 39:519–545, 1992.
8. O. Gotoh. An improved algorithm for matching biological sequences. *J. Mol. Biol.*, 162:705–708, 1982.
9. O. Gotoh. 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, 1996.
10. B. Göttgens, L. Barton, J. Gilbert, A. Bench, M. Sanchez, S. Bahn, S. Mistry, D. Grafham, A. McMurray, M. Vaudin, E. Amaya, D. Bentley, and A. Green. Analysis of vertebrate SCL loci identifies conserved enhancers. *Nature Biotechnology*, 18:181–186, 2000.
11. D. Joseph, J. Meidanis, and P. Tiwari. Determining DNA sequence similarity using maximum independent set algorithms for interval graphs. *Lecture Notes in Computer Science*, 621:326–337, 1992.
12. A. Krause, P. Nicodème, E. Bornberg-Bauer, M. Rehmsmeier, and M. Vingron. Www access to the systers protein sequence cluster set. *Bioinformatics*, 15:262–263, 1999.

13. C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, and J. C. Wootton. Detecting subtle sequence signals: a gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–14, 1993.
14. W. Miller. So many genomes, so little time. *Nature Biotechnology*, 18:148–149, 2000.
15. B. Morgenstern. DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, 15:211–218, 1999.
16. B. Morgenstern. A space-efficient algorithm for aligning large genomic sequences. *Bioinformatics*, 16:948–949, 2000.
17. B. Morgenstern, A. W. M. Dress, and T. Werner. Multiple DNA and protein sequence alignment based on segment-to-segment comparison. *Proc. Natl. Acad. Sci. USA*, 93:12098–12103, 1996.
18. B. Morgenstern, K. Frech, A. W. M. Dress, and T. Werner. DIALIGN: finding local similarities by multiple sequence alignment. *Bioinformatics*, 14:290–294, 1998.
19. B. Morgenstern, K. Hahn, W. R. Atchley, and A. W. M. Dress. Segment-based scores for pairwise and multiple sequence alignments. In J. Glasgow, T. Littlejohn, F. Major, R. Lathrop, D. Sankoff, and C. Sensen, editors, *Proceedings of the Sixth International Conference on Intelligent Systems for Molecular Biology*, pages 115–121, Menlo Parc, CA, 1998. AAAI Press.
20. B. Morgenstern, J. Stoye, and A. W. M. Dress. Consistent equivalence relations: a set-theoretical framework for multiple sequence alignment. Materialien und Preprints 133, University of Bielefeld, 1999.
21. S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.
22. C. Notredame and D. Higgins. SAGA: sequence alignment by genetic algorithm. *Nucleic Acids Research*, 24:1515–1524, 1996.
23. W. R. Pearson and D. J.Lipman. Improved tools for biological sequence comparison. *Proc. Natl. Acad. Sci. USA*, 85:2444–2448, 1988.
24. T. F. Smith and M. S. Waterman. Comparison of biosequences. *Advances in Applied Mathematics*, 2:482–489, 1981.
25. J. Stoye. Multiple sequence alignment with the divide-and-conquer method. *Gene*, 211:GC45–GC56, 1998.
26. J. D. Thompson, D. G. Higgins, and T. J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Research*, 22:4673–4680, 1994.
27. J. D. Thompson, F. Plewniak, and O. Poch. BAliBASE: A benchmark alignment database for the evaluation of multiple sequence alignment programs. *Bioinformatics*, 15:87–88, 1999.
28. J. D. Thompson, F. Plewniak, and O. Poch. A comprehensive comparison of protein sequence alignment programs. *Nucleic Acids Research*, 27:2682–2690, 1999.
29. J. D. Thompson, F. Plewniak, J.-C. Thierry, and O. Poch. DbClustal: rapid and reliable global multiple alignments of protein sequences detected by database searches. *Nucleic Acids Research*, 28:2919–2926, 2000.
30. M. Vingron and P. Argos. Motif recognition and alignment for many sequences by comparison of dot-matrices. *J. Mol. Biol.*, 218(1):33–43, 1991.
31. M. Vingron and P. Pevzner. Multiple sequence comparison and consistency on multipartite graphs. *Advances in Applied Mathematics*, 16:1–22, 1995.
32. J. W. Wilbur and D. J. Lipman. The context dependent comparison of biological sequences. *SIAM J. Appl. Math.*, 44:557–567, 1984.