Divide-and-conquer multiple alignment with segment-based constraints

Michael Sammeth[1,*], *Burkhard Morgenstern*[2] *and Jens Stoye*[1]

[1]*Bielefeld University, Department of Genome Informatics, Technical Faculty, P.O. 10 01 31, 33594 Bielefeld, Germany and* [2]*University of Göttingen, Department of Bioinformatics, Institute of Microbiology and Genetics, Goldschmidtstr. 1, 37077 Göttingen, Germany*

Received on March 17, 2003; accepted on June 9, 2003

ABSTRACT
A large number of methods for multiple sequence alignment are currenty available. Recent benchmarking tests demonstrated that strengths and drawbacks of these methods differ substantially. *Global* strategies can be outperformed by approaches based on *local* similarities and vice versa, depending on the characteristics of the input sequences. In recent years, *mixed* approaches that include both global and local features have shown promising results. Herein, we introduce a new algorithm for multiple sequence alignment that integrates the global *divide-and-conquer* approach with the local *segment-based* approach, thereby combining the strengths of those two strategies.
Contact: micha@sammeth.net

1 INTRODUCTION

Automatic generation of multiple alignments is a central task of computational biology. Although diverse methods are now available, no final solution applicable in all possible alignment situations has been found Notredame (2002). Traditionally, there exist two opposed strategies of alignment construction, one creating *global* alignments and the other one detecting *local* similarities among the input sequences.

For global alignment, *simultaneous* approaches create alignments by synchronising the information of all input sequences in a k-dimensional lattice. Although highly elaborated algorithms have been developed to narrow regions of interest within this lattice Gupta (1995); Tönges *et al.* (1996), these approaches are computationally expensive so that their application is limited. For this reason, alternative approaches have been developed where the multiple alignment problem is reduced to a series of pairwise *profile* alignments Feng and Doolittle (1987); Higgins and Sharp (1988); Taylor (1988); the most popular of these *progressive* methods is CLUSTAL W Thompson *et al.* (1994). However, a serious drawback of this technique is that the resulting multiple alignments crucially depend on the *order* in which the profile alignments are carried out.

To cope with more locally related sequence sets, a number of alternative approaches have been proposed that focus on locally related segments of the sequences Depiereux *et al.* (1997); Morgenstern *et al.* (1996); Schuler *et al.* (1991); Vingron and Argos (1991). These approaches are superior to more traditional strategies in situations where large gaps need to be inserted into the alignment and for data sets that are evolutionarily distantly related. However, they may be outperformed by global methods where sequence sets are related over their entire length Lassmann and Sonnhammer (2002); Thompson *et al.* (1999).

Obviously, it is highly desirable to have alignment algorithms performing well on both, globally and locally related sequences. Notredame *et al.* proposed an approach where both, local and global alignment information, is pairwisely preprocessed and extended to the multiple context in a heuristic solution of the maximum weight trace problem Kececioglu (1993). Biasing those pre-processed similarities improved the results of standard progressive alignment, and the resulting procedure has been implemented in the program T-COFFEE Notredame *et al.* (2000). Moreover, Myers *et al.* developed an algorithm for progressive multiple alignment with *constraints* Myers *et al.* (1996). Herein, we introduce an algorithm that performs simultaneous multiple alignment under constraints given by pre-calculated local sequence similarities. In our implementation, we combine the global divide-and-conquer algorithm DCA Stoye (1998) with the local segment-based program DIALIGN Morgenstern (1999). We evaluate this mixed method and compare its results to both of the native protocols and to other successfull alignment methods (i.e. T-COFFEE and CLUSTAL W).

[*]To whom correspondence should be addressed.

2 TECHNICAL BACKGROUND

A *global alignment* of a family of k sequences $S = (s_1, s_2, \ldots, s_k)$ over a finite alphabet Σ can be defined as a $k \times m$ matrix A with entries in an extended alphabet $\Sigma^* = \Sigma \cup \{-\}$, such that ignoring the blank characters, the pth row reproduces sequence s_p and there is no column consisting exclusively of blanks. A maximal run of adjacent blank characters in a row is called a *gap*. Further, an alignment A can be represented as a source-to-sink path in a k-dimensional *alignment graph*.

In contrast, a *local* alignment aligns only *substrings* of the input sequences. Most frequently, local alignments are used in pairwise sequence comparison ($k = 2$), especially for database searching. A special type of local alignments is the *gap-free* pairwise alignment, by which two substrings of equal length are matched. Note that these substrings are usually expected to share a certain degree of similarity but, in general, they are not required to be identical. Ungapped local alignments between two sequences are sometimes called *fragments* or *diagonals*. Moreover, such gap-free pairwise alignments are also used in the context of multiple sequence comparison. Here, a fragment f is uniquely determined by the involved sequences s_p and s_q, the starting points i and j of the substrings and the fragment length ℓ. Therefore, we will use the shorthand notation $f = ([p, i], [q, j], \ell)$.

Before we will introduce our new fragment-based divide-and-conquer global alignment algorithm, we outline both of the strategies involved, multiple alignment construction based on pairwise fragments and the divide-and-conquer algorithm for simultaneous multiple alignment.

2.1 Segment-based multiple alignment

In the segment-based program DIALIGN, a *weighting* function is defined on the set of all possible fragments, and the program tries to find a *consistent* collection of fragments with maximum total weight Morgenstern (1999). A set F of fragments is called *consistent* if there exists a *global* alignment A_F such that all segment pairs $f \in F$ are aligned by A_F; in this case, we say that A_F *realizes* F. A non-consistent set of fragments is shown in Figure 1.

For multiple alignment, DIALIGN integrates fragments *greedily* into a consistent set F that defines an alignment of the input sequences. To check if a fragment f can be included into F, the algorithm uses so-called *transitivity frontiers*, a data structure first introduced by Abdeddaïm Abdeddaïm (1997). Let us consider the set X of all *positions* x in a sequence family $S = (s_1, s_2, \ldots, s_k)$ where position $x = [p, i]$ corresponds to the ith character in sequence s_p. A collection F of fragments induces a *quasi-partial order relation* $\preceq_F$ on the set X where $x \preceq_F y$ means that x is to the left-hand side or in the same column as y in *every* alignment A_F that realizes F. Formally, the transitivity frontiers are defined as

$$Pred_F(x, p) = \max\{j : [p, j] \preceq_F x\},$$

$$Succ_F(x, p) = \min\{j : x \preceq_F [p, j]\}.$$

In other words, $Pred_F(x, p)$ ($Succ_F(x, p)$) is the index of the right-most (left-most) character in sequence s_p that is to the left (right) of x with respect to the relation $\preceq_F$. These frontiers can be used to decide which residues of the sequences are still alignable without leading to inconsistencies with F Abdeddaïm and Morgenstern (2001), see Figure 2. If the referred set of fragments is obvious, we will drop the index F in $Pred_F(x, p)$ and $Succ_F(x, p)$.

2.2 Divide-and-conquer hyperspace alignment

For a given family of input sequences $S = (s_1, s_2, \ldots, s_k)$, the *divide-and-conquer approach* to simultaneous multiple sequence alignment (DCA) Tönges *et al.* (1996) firstly splits each sequence $s_p \in S$ at a *cut position* c_p. In order to obtain good cut positions, Tönges *et al.* Tönges *et al.* (1996) suggest to use pairwise matrices $C_{p,q}$ for all $s_p, s_q \in S$ that store for each cell (i, j) of the dynamic-programming matrix the *additional alignment costs* that are incurred if the alignment graph is *forced* to leave the optimum and pass the cell (i, j). These matrices are calculated using standard *dynamic programming* procedures. Different heuristics have been described to find families of cut positions $(c_1, c_2, \ldots, c_k)$ that minimise the additional costs in all sequence pairs Stoye (1998). The cutting procedure is recurred until a certain *stop size* of the cut sequences is no longer exceeded. Then, the obtained *atomic* subsequences are aligned optimally and the results are concatenated to form a complete global alignment A.

Within the DCA protocol, the optimal alignment of the atomic sets of subsequences is achieved by applying the *simultaneous alignment* strategy as described for the MSA algorithm Gupta (1995). The latter applies a simultaneous alignment algorithm which basically conquers a k-dimensional (*hyperspace*) lattice using the forward dynamic programming technique concurrently on all sequences in S to find an optimal multiple alignment according to the scoring function. However, heuristics are used to reduce the search effort, e.g. the *Carrillo-Lipman* heuristics Carrillo and Lipman (1988): a progressive alignment is pre-computed for the input set to derive an upper bound on the alignment costs in all projected sequence pairs. Afterwards, pairwise matrices of *total alignment costs* $D_{p,q}$, which are again calculated by dynamic programming procedures, are applied to exclude projections of hyperspace cells from the optimal solution. The final dynamic programming procedure conquering the hyperspace lattice then is sped up by a branch-and-bound procedure skipping all cells which contain an excluded pairwise projection.

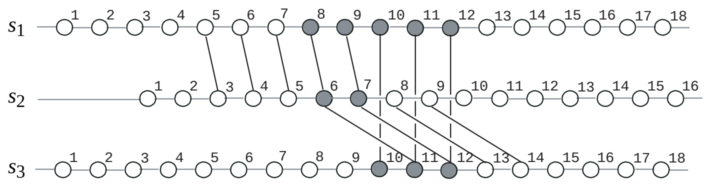

Fig. 1. A set of fragments $f_1, \ldots, f_N$ may be *inconsistent*, i.e., it may not be possible to include all fragments simultaneously in a single multiple alignment. In our example, the three fragments $f_1 = ([1, 5], [2, 3], 5)$, $f_2 = ([2, 6], [3, 11], 4)$ and $f_3 = ([1, 10], [3, 10], 3)$, shown as lines between the aligned indices, would lead to contradicting alignment of the positions marked in grey in a global multiple alignment.

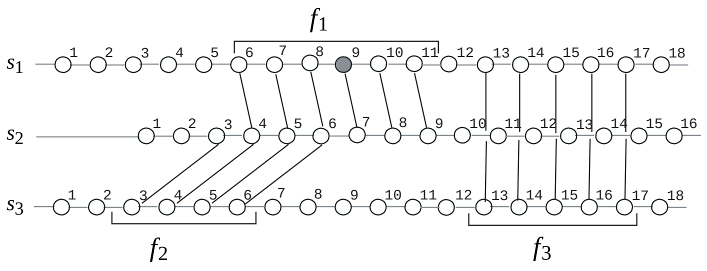

Fig. 2. Example for transitivity frontiers of position [1,9] (marked grey) enforced by the set of fragments $F = \{f_1, f_2, f_3\}$ with $f_1 = ([1, 6], [2, 4], 6)$, $f_2 = ([2, 3], [3, 3], 4)$, $f_3 = ([1, 13], [3, 13], 5)$. For s_3, *Pred*$([1, 9], 3) = 6$ and *Succ*$([1, 9], 3) = 13$ are induced. In the special case of aligned positions, transitivity frontiers coincide, for example in s_2, a single position is specified by *Pred*$([1, 9], 2) =$ *Succ*$([1, 9], 2) = 7$.

3 THE ALGORITHM

On a high level, our algorithm proceeds as follows: in a first step, we apply the segment-based alignment in order to obtain a consistent set F of fragments representing a framework for further refinement. These fragments—or a suitable subset of them—are used as *constraints* in the second step of the algorithm. Here, we apply the divide-and-conquer method to complete the alignment for those regions that are not aligned by the fragments in F. More precisely, divide-and-conquer computes a (near-)optimal multiple alignment A under the additional constraint that A realises the set F. To control the influence of the fragments in F on the final alignment, we apply a cutoff threshold T and accept only those fragments $f \in F$ that have *overlap weights* exceeding T (see Morgenstern *et al.* (1996) for a definition of these overlap weights). With $T = 0$, the entire set F is used as constraints, and the final alignment is therefore just a minor modification of the DIALIGN alignment. With higher cutoff values, the influence of the segments is reduced, and the resulting output alignments are more similar to what the original divide-and-conquer algorithm would return. If T is large enough to exclude *all* fragments $f \in F$, we obtain exactly the DCA alignment.

While the computation of the fragments can be performed exactly as in the DIALIGN algorithm (or any other procedure that computes consistent local similarities from a family of sequences), three modifications to the divide-and-conquer algorithm are necessary: (1) in the divide step, the cut positions selected to divide the sequences need to be in accordance with our fragment-induced constraints; and in each conquer step, (2a) the heuristic multiple alignment used to calculate the Carrillo-Lipman bounds and (2b) the simultaneous multiple alignment procedure carried out to compute an alignment A_F within these bounds must respect the constraints given by F. These modifications are detailed in the following two subsections.

3.1 The *divide* step with constraints

Our first modification to the original DCA algorithm concerns the way we compute the *additional-cost* matrices that are used to determine cut positions for the sub-alignments. The consistency constraints prevent certain pairs of residues from being aligned to each other. The corresponding graphs in the alignment matrices must therefore be *masked*. To this end, we simply assign infinite additional costs to those edges. Given a sequence pair (s_p, s_q), we need to know those positions j in s_q that are alignable with position $[p, i]$ without leading to inconsistencies; this information must be available for each position $[p, i]$ in sequence s_p and vice versa for each position $[q, j]$ in sequence s_q.

There are two possible scenarios (cf. Figs 2 and 3): the first scenario is that position $[p, i]$ is already (directly or indirectly) aligned with some position $[q, j]$ by the fragments in F. This is the case if and only if we have $Pred([p, i], q) = Succ([p, i], q) = j$. In this case, there is only *one* possible edge in the respective pairwise alignment matrix leading to cell (i, j), namely the diagonal edge coming from position $(i - 1, j - 1)$, see Figure 3, left. The two edges coming from $(i - 1, j)$ and $(i, j - 1)$ have to be excluded—together with all edges to the left and above this edge, as they would correspond to a gap character aligned to position $[p, i]$, in contradiction to the given fragments that impose that $[p, i]$ is aligned with $[q, j]$. The second scenario is that $[p, i]$ and $[q, j]$ are not (yet) aligned, which is the case if and only if $Pred([p, i], q) < Succ([p, i], q)$ holds (Fig. 3, center). In this case, all positions between $Pred([p, i], q) + 1$ and $Succ([p, i], q) - 1$ can be aligned with $[p, i]$ without leading to inconsistencies. For $Pred([p, i], q) + 1 \leq j \leq Succ([p, i], q) - 1$ (Fig. 3, right), the cell (i, j) in the dynamic programming matrix can be reached from all three possible positions. In addition, cell $(i, Pred([p, i], q))$ can be reached from above (but not from the left or from the top-left) and cell $(i, Succ([p, i], q))$ can be reached from the left (but not from above or from the top-left), see Figure 3, center and right. After this masking procedure, the original procedure used in DCA can be employed to identify suitable cut positions for the *divide* step.

3.2 The *conquer* step with constraints

In the Carrillo-Lipman approach, all optimal pairwise alignments as well as a heuristic multiple alignment of the input sequences are computed in order to determine *boundaries* for the pairwise *projections* of an optimal multiple alignment to the respective pairwise comparison matrices Carrillo and Lipman (1988). In our constrained scenario, we need to impose the consistency constraints both to the optimal pairwise alignments and to the heuristic multiple alignment. The pairwise alignments are calculated as explained in Section 3.1. For the heuristic multiple alignment, where we use a progressive approach with *profile alignments* described in Gupta (1995), in addition to alignments of individual sequences, a very similar consistency-constrained approach can be applied, too. To determine which positions are alignable between two profiles, the transitivity frontiers of all involved sequence pairs are considered. The alignable regions for the profiles are then given as the *intersections* of the alignable regions for the individual sequence pairs.

Once the pairwise alignments and the heuristic multiple alignment are computed, restrictions for the *projections* of an optimal multiple alignment to all pairwise comparison matrices can be calculated. In our approach, we have additional constraints derived from our transitivity frontiers. The regions allowed for the projections to the pairwise matrices are simply given as the intersections between the regions calculated using the Carrillo-Lipman boundaries and the regions defined by our transitivity frontiers. An optimal multiple alignment realising F is finally computed in the space that is defined by these combined pairwise restrictions, similar to the original Carrillo-Lipman approach.

4 RESULTS AND DISCUSSION

To evaluate the performance of our new algorithm, we benchmarked it against DIALIGN, DCA, CLUSTAL W and T-COFFEE. For DCA, we used a re-implementation that employs a more stable version of MSA. For all programs, we used their default parameter settings together with the BLOSUM-62 substitution matrix. Our algorithm was implemented in JAVA and the program was tested with a hotspot capable runtime environment (JDK1.4.1 with a maximum of 3 GB heap memory). The main benchmarks were performed on a SUN workstation (SUN Fire 880, 750 MHz, 32 GB main memory). Data for benchmarking was derived from the reference alignments in BAliBASE, version 1 Thompson *et al.* (1999), where reference alignments are sorted into five groups according to the characteristics of the input, respectively. Within the first three groups, the sequences of the input share a variant degree of global affinity, but similarity drops from group 1 to group 3. Group 4 contains alignments with N/C-terminal extensions, while in group 5 sequences with internal insertions have been collected. To compare the computed alignments to the BAliBASE benchmark alignments, we used the program `aln_compare` written by C. Notredame (personal communication).

Table 1 summarizes the main results. For each group of alignments from the BAliBASE data set, the average percentage identity with the reference alignments is given, taking into account structural columns as annotated in BAliBASE. A basic version of our program denoted by *Mixed* $T = 0$ in Table 1 uses the entire set of fragments returned by DIALIGN as constraints for the divide-and-conquer alignment. In addition, we performed runs where the cutoff value T was set to higher values, such that only fragments with overlap weights above T were used. For comparison, we also ran two popular multiple alignment programs from the literature, CLUSTAL W Thompson *et al.* (1994) and T-COFFEE Notredame *et al.* (2000).

As can be seen from Table 1, the strengths of DCA and DIALIGN differ substantially. The mixed approach shows characteristics of both strategies, refining the segment-based local alignment computed by DIALIGN with the divide-and-conquer hyperspace alignment stragegy (DCA). Problems arise in group 2 and group 3 where neither DIALIGN can provide good local similarities,

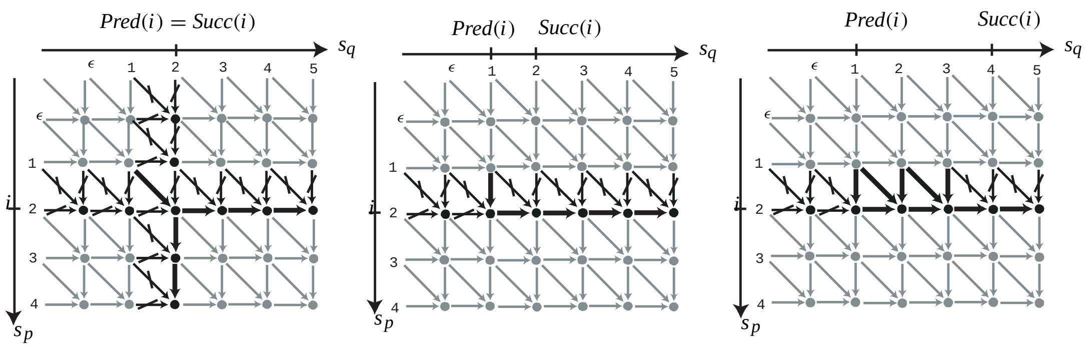

Fig. 3. Three examples of transitivity frontiers for $i = 2$ in a two dimensional dynamic programming matrix of s_p and s_q. Black dots represent cells within the matrix of which incoming edges are affected by the transitivity frontiers considered here.

Table 1. Percent identity of the results produced by the various algorithms with the reference alignments given by BAliBASE (according to the column score of trusted regions). The alignment count is given in parentheses after each group in the top row

Algorithm	**Group 1 (83)**	**Group 2 (23)**	**Group 3 (12)**	**Group 4 (13)**	**Group 5 (11)**
DCA	79.27	29.47	37.47	58.81	78.93
Mixed $T = 10$	76.03	26.06	35.71	71.52	72.45
Mixed $T = 7$	76.11	24.66	33.84	73.74	80.54
Mixed $T = 5$	76.26	25.49	34.04	74.82	85.08
Mixed $T = 3$	74.39	24.86	35.73	78.93	83.74
Mixed $T = 0$	73.07	25.39	35.49	75.29	80.38
DIALIGN	71.95	25.13	35.10	74.66	80.38
CLUSTAL W	79.53	32.91	48.72	74.02	67.84
T-COFFEE	76.86	36.89	49.94	81.28	86.25

nor does the scoring function maximized by DCA yield results close to the reference alignments. However, in group 1 and in the last two groups, the best scoring result of the mixed strategy is very close to the T-COFFEE results. Note that the values given are arithmetic averages over all alignments in a group such that single alignments within a group can score even better than with T-COFFEE. Because of the global affinity within group 1, the global strategies perform very good there (see CLUSTAL W, DCA). In contrast, exclusively global methods perform rather poorly in group 4 and group 5, whereas the mixed strategy can successfully integrate the local fragments into the global alignment (i.e. it scores better than any of the underlying strategies alone).

Concerning the influence of the cutoff parameter on the quality of the resulting alignment, it can be seen from Table 1 that the best value for T depends on the characteristics of the input data. In the first three groups of BAliBASE where DCA is superior to DIALIGN, a high cutoff prevents spurious fragments from deteriorating the final alignment and increases the quality of the program output. For groups 4 and 5, however, DIALIGN fragments tend to improve the alignment produced by DCA. Here, a lower cutoff leads to better results.

Table 2 reports on the running time for the programs we analyzed. Alignments that did not terminate within 24 hours were considered as *failures*. The reason for this phenomenon is that, even with efficient heuristics, the search for high-quality cut positions and the subsequent hyperspace alignment may be time-consuming for large or badly matching sequence families. Our results show that if input sequences are easy to align, the mixed protocol takes slightly more time than the two underlying algorithms together. This is because additional CPU time is necessary to mask all matrices used in DCA according to the set of fragments found by DIALIGN. However, the time spent for this purpose quickly pays off for input sequences that are difficult to align such as in groups 2–5. Here all mixed results show less failures and are aligned faster than with the original DCA algorithm. Fragments can, of course, slow down the search for optimal cut points if they are not in agreement with the global optimal alignment (see failure counts in Table 2, Mixed $T = 10$). In general, however, the fragments from the local alignment reduce the part of the hyperspace that needs to be considered by DCA, thereby speeding up the divide-and-conquer

Table 2. Time consumption in seconds and number of failed runs (in parentheses) for the various programs summed up for each BAliBASE group. The total number of alignments in each group is given in parentheses in the top row

Algorithm	Group 1 (83)		Group 2 (23)		Group 3 (12)		Group 4 (13)		Group 5 (11)	
DCA	136.33	(0)	119430.51	(1)	307753.98	(5)	9918.90	(2)	42252.48	(2)
Mixed $T = 10$	766.02	(0)	8942.64	(1)	1909.89	(1)	4427.80	(1)	42411.52	(1)
Mixed $T = 7$	680.62	(0)	8634.20	(0)	1475.39	(2)	1969.15	(1)	42175.50	(1)
Mixed $T = 5$	445.24	(0)	3435.06	(0)	1360.25	(2)	459.20	(1)	42051.03	(0)
Mixed $T = 3$	487.88	(0)	3014.28	(0)	1360.90	(2)	324.16	(1)	17202.66	(0)
Mixed $T = 0$	435.00	(0)	1652.18	(0)	1279.20	(0)	289.98	(0)	276.57	(0)
DIALIGN	136.67	(0)	525.97	(0)	386.45	(0)	141.51	(0)	94.49	(0)
CLUSTAL W	140.19	(0)	445.47	(0)	136.28	(0)	71.25	(0)	50.67	(0)
T-COFFEE	447.68	(0)	4614.28	(0)	2485.24	(0)	913.01	(0)	550.33	(0)

Table 3. Column scores and running time (in parentheses) for different algorithm runs in group 5 of the BAliBASE

data set	Mixed $T = 5$		Mixed $T = 3$		Mixed $T = 0$		T-COFFEE	
1eft	71.4	(5.03)	71.4	(5.30)	71.4	(5.66)	71.4	(15.94)
1ivy	94.7	(21.19)	92.8	(21.39)	77.8	(20.85)	75.6	(20.40)
1qpg	100.0	(73.35)	100.0	(74.60)	100.0	(73.82)	100.0	(131.08)
1thm1	66.7	(19.83)	66.7	(18.43)	55.6	(10.83)	83.3	(37.49)
1thm2	76.7	(4.01)	76.7	(3.71)	88.3	(4.03)	88.3	(12.85)
2cba	100.0	(11.23)	100.0	(9.35)	100.0	(9.22)	96.7	(17.52)
kinase1	93.5	(4.87)	80.6	(4.52)	80.6	(5.83)	91.0	(78.98)
kinase2	66.7	(76.55)	66.7	(79.93)	66.7	(31.64)	100.0	(5.37)
kinase3	81.2	(83.93)	81.2	(94.08)	81.2	(77.56)	80.6	(6.23)
S51	85.0	(41744.78)	85.0	(16884.04)	62.6	(32.90)	88.9	(62.90)
S52	100.0	(6.28)	100.0	(7.32)	100.0	(4.24)	72.9	(161.57)

approach and causing fewer fails. Obviously, this time gain is reduced if more of the fragments are filtered out. Nevertheless, in most instances the mixed strategy is still faster than T-COFFEE which needs additional CPU time to preprocess and bias local information of the input and therefore takes about ten times the running time of CLUSTAL W. To our knowledge this is the first time that a simultaneous alignment strategy is about as time efficient as progressive heuristics. Moreover, note that the mixed strategy is implemented in JAVA as opposed to CLUSTAL W and T-COFFEE that are written in C, such that there is probably an even bigger algorithmic time gain than reflected by the time measurements shown in Table 2.

Finally, Table 3 gives an overview of the quality and running time for all 11 individual alignments of BAliBASE group 5. In the specific alignment runs, a general trend of quality and time gain for different cutoff values is hard to observe. However, in a few cases the time effort is sensitively triggered by T (e.g. the running time for S51 runs can be reduced from 11.6 to 4.7 hours by increasing the number of fragments from $T = 5$ to $T = 3$ while the score remains the same). Although both protocols inherit from global and local alignment methods, results show that the strengths of mixed DIALIGN–DCA strategy differ from the ones of T-COFFEE (e.g. the kinase2 set is aligned correctly by T-COFFEE while the mixed protocol just reaches 2/3 identity with the reference, whereas for the S52 set the situation is the reverse). Thus, carefully chosen fragments are essential in combining local and global alignment strategies.

As could be demonstrated by the BAliBASE benchmark results, the mixed alignment strategy successively combines the strengths of DIALIGN (local) and DCA (global) multiple alignment. On the one side, the mixed method takes full advantage of the hyperspace alignment, while it reduces the computational time (and space) requirements necessary for this method (see Table 2). In most reference groups, the average results can reach a value close to the T-COFFEE algorithm (see Table 1), and they are generally faster calculated. Single alignments within a group of the BAliBASE reference set are even outperforming results of the T-COFFEE algorithm (see Table 3).

Furthermore, different values empirically tested for the cutoff T of fragment scores were found to show peaks in alignment quality for each group in the BAliBASE reference. Hence, in further work we will concentrate on replacing the naive threshold-based fragment filter by a weighting scheme that biases the score of each fragment during the global alignment construction. Our hope is that we can achieve a simultaneous dynamic programming algorithm that considers the extended information of local fragments but is not misled if the similarities found turn out to be incorrect for a global solution.

ACKNOWLEDGEMENTS

We would like to thank Cedric Notredame for providing a reliable program for the comparison of test alignments to the BAliBASE reference alignments.

REFERENCES

Abdeddaïm,S. (1997) Incremental computation of transitive closure and greedy alignment. *Proceedings of 8th Annual Symposium on Combinatorial Pattern Matching (CPM 1997)*, volume 1264 of *LNCS*, pp. 167–179.

Abdeddaïm,S. and Morgenstern,B. (2001) Speeding up the DIALIGN multiple alignment program by using the 'greedy alignment of biological sequences library' (GABIOS-LIB). *Proceedings Journées Ouvertes: Biologie, Informatique, Mathématiques (JOBIM 2000)*, volume 2066 of *LNCS*, pp. 1–11.

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

Depiereux,E., Baudoux,G., Briffeuil,P., Reginster,I., Boll,X.D., Vinals,C. and Feytmans,E. (1997) Match-Box server: a multiple sequence alignment tool placing emphasis on reliability. *CABIOS*, **13**, 249–256.

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

Gupta,S., Kececioglu,J. and Schäffer,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**, 459–472.

Higgins,D. and Sharp,P. (1988) CLUSTAL—a package for performing multiple sequence alignment on a microcomputer. *Gene*, **73**, 237–244.

Kececioglu,J. (1993) The maximum weight trace problem in multiple sequence alignment. *Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching (CPM 1993)*, volume 684 of *LNCS*, pp. 106–119.

Lassmann,T. and Sonnhammer,E. (2002) Quality assessment of multiple alignment programs. *FEBS Lett.*, **529**, 126–130.

Morgenstern,B. (1999) DIALIGN 2: improvement of the segment-to-segment approach to multiple sequence alignment. *Bioinformatics*, **15**, 211–218.

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

Myers,G., Selznick,S., Zhang,Z. and Miller,W. (1996) Progressive multiple alignment with constraints. *J. Comp. Biol.*, **3**, 563–572.

Notredame,C. (2002) Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, **3**, 131–144.

Notredame,C., Higgins,D. and Heringa,J. (2000) T-COFFEE: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

Schuler,G.D., Altschul,S.F. and Lipman,D.J. (1991) A workbench for multiple alignment construction and analysis. *Proteins: Struct. Funct. Genet.*, **9**, 180–190.

Stoye,J. (1998) Multiple sequence alignment with the divide-and-conquer method. *Gene*, **211**, GC45–GC56.

Taylor,W.R. (1988) A flexible method to align large numbers of biological sequences. *J. Mol. Evol.*, **28**, 161–169.

Thompson,J., Higgins,D. and Gibson,T. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Thompson,J., Plewniak,F. and Poch,O. (1999) BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs. *Bioinformatics*, **15**, 87–88.

Thompson,J., Plewniak,F. and Poch,O. (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, **27**, 2682–2690.

Tönges,U., Perrey,S., Stoye,J. and Dress,A. (1996) A general method for fast multiple sequence alignment. *Gene*, **172**, GC33–GC41.

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