BIOINFORMATICS APPLICATIONS NOTE Vol. 16 no. 10 2000
Pages 948–949

A space-efficient algorithm for aligning large genomic sequences

Burkhard Morgenstern

MIPS–Max-Planck-Institut für Biochemic, Am Klopferspitz 18a, 82152 Martinsried, Germany and Aventis Pharma Limited, Rainham Road South, Essex RM10 7XS, UK

Received on March 23, 2000; revised and accepted on June 5, 2000

Abstract

Summary: *In the segment-by-segment approach to sequence alignment, pairwise and multiple alignments are generated by comparing gap-free segments of the sequences under study. This method is particularly efficient in detecting local homologies, and it has been used to identify functional regions in large genomic sequences. Herein, an algorithm is outlined that calculates optimal pairwise segment-by-segment alignments in essentially linear space.*

Availability: *The program is available at the Bielefeld Bioinformatics Server (BiBiServ) at http://bibiserv.techfak.uni-bielefeld.de/dialign/*

Contact: *morgenstern@mips.biochem.mpg.de*

The DIALIGN program constructs pairwise and multiple alignments of nucleic acid and protein sequences from un-gapped segment pairs, each segment from one of the sequences under study, see Morgenstern *et al.* (1996) and Morgenstern (1999). In the literature, such segment pairs are referred to as *fragments*, and an alignment can be defined as a suitable collection of fragments (in previous publications, we used the term *diagonal* instead of *fragment*). Note that, with this definition, fragments of varying length are considered for alignment, and mismatches are allowed in fragments. In order to find 'good' alignments, every possible fragment f is given a non-negative weight score $w(f)$ reflecting the degree of similarity among the two segments, and the overall score of an alignment is then defined as the sum of weights of the fragments it is composed of. Thus, for pairwise alignment, the optimisation problem is to find a *chain* of fragments $f_1 \ll \cdots \ll f_k$ such that the sum $\sum_i w(f_i)$ is maximal. Here, $f_i \ll f_j$ means that, in both sequences, the end positions of f_i are strictly smaller than the respective beginning positions of f_j. In the DIALIGN approach, constructing optimal pairwise alignments in the sense of this fragment-based objective function is also the first step in a greedy procedure for multiple alignment, see Morgenstern (1999) and Abdeddaïm and Morgenstern (2000) for details.

The segment-by-segment alignment method is able to identify functionally important regions even in large genomic sequences, see, for example, Göttgens *et al.* (2000). Here, it is essential to have space-efficient alignment programs. There are well-known algorithms that solve the above fragment-chaining problem. If a set F of allowed fragments is *given*, the problem can be solved in $O(\#F)$ space, see Wilbur and Lipman (1984). Obviously, a term for the sequence length has to be added if, in a first step, the set F is generated, so the complete problem can be solved by standard methods in $O(L + \#F)$ space where L is the length of the longer sequence. Chao and Miller (1995) have proposed an efficient algorithm for the case where *maximal* pairs of *identical* segments are considered and an affine penalty is charged for connecting two fragments. By combining a *sparse* dynamic programming algorithm by Eppstein *et al.* (1992) with the divide-and-conquer method by Hirschberg (1975) and Myers and Miller (1988), they obtained an algorithm that runs in space proportional to the length of the input sequences.

In this note, a rather simple algorithm is outlined that solves the segment-to-segment alignment problem in the case where general gap-free segment pairs are allowed and gaps between fragments are not penalised. For the fragment-weighting function used in the DIALIGN program, the proposed method runs in space essentially proportional to the length of the input sequences. The idea behind our algorithm is as follows. We consider two sequences $\mathbf{a} = a_1 a_2 \ldots a_{L_1}$ and $\mathbf{b} = b_1 b_2 \ldots b_{L_2}$. The comparison matrix is processed in a column-by-column fashion from left to right. With every fragment $f \in F$ that *starts* in the current column i, we associate the *score* of an optimal alignment *ending* in f, together with a pointer to the *predecessor* of f, i.e. to the *second-last* fragment in this alignment. This can be done if the scores of optimal alignments of the *prefixes* $a_1 \ldots a_{i-1}$ and $b_1 \ldots b_j$ are known for all positions $(i-1, j), 1 \leq j \leq L_2$, in column $i-1$,—and if for each of these positions, there is a pointer to the *last* fragment in the respective optimal prefix alignment. A pointer to f is then added to a list

$F_{i'}$ that is associated with the column i' in which f *ends*. Once all fragments *starting* in column i are processed, the scores and last fragments of the optimal prefix alignments are calculated for column i. Here, the respective values from column $i-1$ are used, together with the previously established list F_i of fragments *ending* in column i. After the scores and last fragments of the prefix alignments are calculated for column i, the respective values for column $i-1$ can be deleted. It is easy to see that a fragment $f \in F_i$ needs to be considered for optimal alignment of the sequences **a** and **b** only if f is last fragment in an optimal alignment of some pair of prefixes $a_1 \ldots a_{i-1}$ and $b_1 \ldots b_j, 1 \leq j \leq L_2$. Thus, after column i is processed and pointers to the last fragments of the 'prefix alignments' are set, fragments $f \in F_i$ to which no pointer has been set can be deleted. Once all columns have been processed in this way, a trace-back procedure is carried out in order to retrieve an optimal alignment of the input sequences. At position (L_1, L_2), there is a pointer to the last fragment of an optimal alignment of **a** and **b** which, in turn, has a pointer to the second-last fragment in this optimal alignment, etc.

The following data need to be stored during our alignment procedure: scores of optimal prefix alignments and pointers to their respective last fragments are stored for one column at a time, so this takes $O(L_2)$ space. Next, lists F_i of all fragments $f \in F$ *ending* in column i are generated for $1 \leq i \leq L_1$. Storing these fragments simultaneously would require memory proportional to $\sum_i \#F_i = \#F$, so the worst-case space complexity of our algorithm is $O(L + \#F)$. As explained above, however, once column i has been processed, some of the fragments $f \in F_i$ can be deleted. Since this is done while subsequent lists $F_{i'}, i' > i$, are still being generated not all fragments in F are stored simultaneously, so the *real* memory requirement of our algorithm is $O(L + N_{\max})$ where $N_{\max} \leq \#F$ is the maximum number of fragments that are stored *simultaneously* during the described procedure.

If the length of allowed fragments is bounded by some constant $l_{\max}$, approximately $L_1 \times L_2 \times l_{\max}$ fragments are to be looked at. In DIALIGN, a default value of $l_{\max} = 40$ is used. Since only fragments with positive weight scores are relevant for optimal alignment, the size of the set F depends on the weighting function w. DIALIGN 2 uses a weighting scheme where, even for closely related sequences, most fragments f have weight scores $w(f) = 0$ and can therefore be ignored, see Morgenstern (1999) for details. The number of fragments with positive weight scores depends on the degree of similarity among the input sequences. Therefore, in order to obtain lower and upper estimates for $\#F$ and $N_{\max}$, we aligned extreme dissimilar as well as extreme similar test sequences, namely pairs of *independent* random sequences and pairs of *identical* random sequences. Table 1 contains the results of these test runs. Even for identical sequences, $N_{\max}$ is not much larger than L. Since here almost all fragments are on the main diagonal, $\#F$ and $N_{\max}$ will grow approximately linearly with the sequence length.

Table 1. Number $\#F$ of fragments considered for alignment and maximal number $N_{\max}$ of fragments $f \in F$ that are stored *simultaneously* during the alignment procedure. Test sequences are pairs of *independent* random sequences and random sequences aligned with themselves (identical random sequences). Alignments were performed by comparing segment pairs on the nucleic acid sequence level (`-n`) as well as by translating nucleic acid segments to peptide segments (`-nt`), see Morgenstern *et al.* (1996) for a description of these options

L		Indep. random seq.		Ident. random seq.	
		`-n`	`-nt`	`-n`	`-nt`
20 kb	$\#F$	824	470	580 041	661 373
	$N_{\max}$	212	156	42 239	20 207
100 kb	$\#F$	6687	3612	2 806 642	3 262 718
	$N_{\max}$	1142	556	262 277	90 734

References

Abdeddaïm,S. and Morgenstern,B. (2000) Speeding up the DIALIGN multiple alignment program by using the 'Greedy Alignment of BIOlogical Sequences LIBrary' (GABIOS-LIB). In Caraux,G., Gascuel,O. and Sagot,M.-F. (eds), *Proceedings of the Journées Ouvertes: Biologie, Informatique et Mathematiques (JOBIM)* pp. 1–8.

Chao,K.M. and Miller,W. (1995) Linear-space algorithms that build local alignments from fragments. *Algorithmica*, **13**, 106–134.

Eppstein,D., Galil,Z., Giancarlo,R. and Italiano,G. (1992) Sparse dynamic programming I: linear cost functions. *J. Assoc. Comp. Mach.*, **39**, 519–545.

Göttgens,B., Barton,L.M., Gilbert,J.G.R., Bench,A.J., Sanchez,M.J., Bahn,S., Mistry,S., Grafham,D., McMurray,A., Vaudin,M., Amaya,E., Bentley,D.R. and Green,A.R. (2000) Analysis of vertebrate SCL loci identifies conserved enhancers. *Nat. Biotechnol.*, **18**, 181–186.

Hirschberg,D.S. (1975) A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, **18**, 341–343.

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.

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

Myers,G. and Miller,W. (1988) Optimal alignments in linear space. *CABIOS*, **4**, 11–17.

Wilbur,W.J. and Lipman,D.J. (1984) The context dependent comparison of biological sequences. *SIAM J. Appl. Math.*, **44**, 557–567.