BIOINFORMATICS

Vol. 15 no. 4 1999
Pages 298–304

Calign: aligning sequences with restricted affine gap penalties

Kun-Mao Chao

Department of Computer Science and Information Management, Providence University, Shalu, Taichung, Taiwan

Received on June 8, 1998; revised on November 18, 1998; accepted on January 8, 1999

Abstract

Motivation: *Given a genomic DNA sequence, it is still an open problem to determine its coding regions, i.e. the region consisting of exons and introns. The comparison of cDNA and genomic DNA helps the understanding of coding regions. For such an application, it might be adequate to use the restricted affine gap penalties which penalize long gaps with a constant penalty.*

Results: *Several techniques developed for solving the approximate string-matching problem are employed to yield efficient algorithms for computing the optimal alignment with restricted affine gap penalties. In particular, efficient algorithms can be derived based on the suffix automaton with failure transitions and on the diagonalwise monotonicity of the cost tables. We have implemented the above methods in C on Sun workstations running SunOS Unix. Preliminary experiments show that these approaches are very promising for aligning a cDNA sequence with a genomic DNA sequence.*

Availability: *Calign is available free of charge by anonymous ftp at: iubio.bio.indiana.edu, directory: molbio/align, files: calign.driver.c calign.c. Another URL reference for the files is http://iubio.bio.indiana.edu/soft/molbio/align/calign.c.*

Contact: *kmchao@csim.pu.edu.tw*

Introduction

The main aim of the Human Genome Project is to determine the roughly three billion nucleotides of the human genome, which encodes all the genetic information of a human, by the year 2005. These data, essentially a gigantic string of the four letters A, C, G and T, can help us to understand and eventually treat many of the >4000 genetic diseases that afflict mankind. However, the data will be useless unless proper methods are developed to interpret the information encoded. A variety of sequence comparison problems are motivated by the analysis of DNA sequences (Waterman, 1984; Wilbur and Lipman, 1984; Pearson and Lipman, 1988; Altschul *et al*., 1990, 1997; Hardison *et al*., 1994; Chao and Miller, 1995). An example is the problem of computing the similarity between two sequences, such as the cost of converting one into another using insertions, deletions and substitutions of letters (Needleman and Wunsch, 1970; Gotoh, 1982).

Affine gap penalties are generally considered appropriate for aligning DNA and protein sequences (Myers and Miller, 1988; Gotoh, 1990). ('Affine' means that a gap of length k is penalized $\alpha + k\beta$, i.e. it costs α to open up a gap plus β for each symbol in the gap.) For certain applications, such as aligning a cDNA sequence with a genomic DNA sequence, it might be adequate to penalize each long gap with a constant penalty. In this paper, we consider a variant of affine gap penalties, called restricted affine gap penalties (Huang, 1994), which encourages long internal gaps in the shorter sequence by imposing a constant penalty for those long gaps in the shorter sequence. In particular, efficient algorithms are proposed for dealing with the situation where the shorter sequence is very similar to a conjunction of some contiguous regions of the longer sequence.

Given a genomic DNA sequence, it is still an open problem to determine its coding regions, i.e. the region consisting of exons and introns. The comparison of cDNA and genomic DNA helps the understanding of coding regions (Gelfand *et al*., 1996; Sze *et al*., 1998). Besides, it can be used to correct the sequencing errors (Daniels *et al*., 1992; Plunkett *et al*., 1993; Chao *et al*., 1995). Furthermore, imagine that both human and mouse genomes have been sequenced and all mouse genes have been experimentally confirmed. With the assistance of the alignments of the human genomic sequences and the mouse cDNA sequences, the mouse genes might be used to predict human genes, or at least verify the prediction of human genes (Sze and Pevzner, 1997).

Several techniques developed for solving the (approximate) string-matching problem (Landau *et al*., 1988; Baeza-Yates and Gonnet, 1994; Crochemore *et al*., 1994; Dermouche, 1995) can be utilized to yield efficient algorithms for computing the optimal alignment with restricted affine gap penalties. In particular, efficient algorithms can be derived based on the suffix automaton with failure transitions and on the diagonalwise monotonicity of the cost tables (Myers, 1986; Myers and Miller, 1989; Ukkonen and Wood, 1993; Chao *et al*., 1997). Moreover, the heuristic method based on counting the number of q-grams (Kim and Shawe-Taylor, 1992; Ukkonen, 1992) can be used to locate the interval in the longer sequence that should be aligned with the shorter sequence.

Table 1. The weights and aligned pairs associated with edges of $G_{A,B}$

Edge	Weight	Aligned pair	Range
$(i-1,j)_D \rightarrow (i,j)_D$	β	$\begin{bmatrix} a_i \\ - \end{bmatrix}$	$i \in [1,M]$ and $j \in [0,N]$
$(i-1,j)_S \rightarrow (i,j)_D$	$\alpha + \beta$	$\begin{bmatrix} a_i \\ - \end{bmatrix}$	$i \in [1,M]$ and $j \in [0,N]$
$(i,j-1)_I \rightarrow (i,j)_I$	β	$\begin{bmatrix} - \\ b_j \end{bmatrix}$	$i \in [0,M]$ and $j \in [1,N]$
$(i,j-1)_S \rightarrow (i,j)_I$	$\alpha + \beta$	$\begin{bmatrix} - \\ b_j \end{bmatrix}$	$i \in [0,M]$ and $j \in [1,N]$
$(i-1,j-1)_S \rightarrow (i,j)_S$	$\sigma(a_i,b_j)$	$\begin{bmatrix} a_i \\ b_j \end{bmatrix}$	$i \in [1,M]$ and $j \in [1,N]$
$(i,j)_D \rightarrow (i,j)_S$	0	None	$i \in [0,M]$ and $j \in [0,N]$
$(i,j)_I \rightarrow (i,j)_S$	0	None	$i \in [0,M]$ and $j \in [0,N]$

We used the new program to align the complete human genomic sequences from GenBank which have related proteins from other species (Gelfand *et al*., 1996; Schuler *et al*., 1996). Preliminary tests show that the alignments accurately locate almost all the coding regions. To demonstrate its strength for dealing with long sequences, we also aligned some UniGene clusters with the genomic sequences.

System and methods

The program described in this paper is written in C and was developed on Sun workstations running SunOS Unix. The code is portable and has been implemented on IBM-PC and Macintosh personal computers.

Algorithm

Preliminaries

Let $A = a_1a_2 \ldots a_M$ and $B = b_1b_2 \ldots b_N$ be two sequences of length M and N, respectively, where without loss of generality $N \geq M$. An alignment of A and B is obtained by introducing dashes into the two sequences such that the lengths of the two resulting sequences are identical and no column contains two dashes. The following define affine gap penalties, which are generally considered appropriate for aligning DNA and protein sequences. Let Σ denote the input symbol alphabet. A non-negative cost $\sigma(a,b)$ is defined for each $(a,b) \in \Sigma \times \Sigma$. A gap of length k costs $\alpha + k\beta$. The cost of an alignment is the sum of σ cost of all columns with no dashes plus the penalties of the gaps.

It is helpful to think of an alignment as a path in the alignment graph, $G_{A,B}$, defined as follows. $G_{A,B}$ is a directed graph with $3(M + 1)(N + 1)$ nodes, denoted $(i,j)_D$, $(i,j)_I$ and $(i,j)_S$, where $i \in [0,M]$ and $j \in [0,N]$. Table 1 depicts all the edges in $G_{A,B}$.

Let s denote $(0,0)_S$ and t denote $(M,N)_S$. A path is normal if, and only if, it does not contain subpaths of the form $(i-1,j)_D \rightarrow (i-1,j)_S \rightarrow (i,j)_D$ or $(i,j-1)_I \rightarrow (i,j-1)_S \rightarrow (i,j)_I$. It can be shown that alignments of A and B are in one-to-one correspondence with normal s-t paths. Furthermore, define the cost of an s-t path P, denoted as $Cost(P)$, to be the sum over the weights of its edges. $Cost(P)$ is exactly the cost of the alignment corresponding to P.

Let $D(i,j)$, $I(i,j)$ and $S(i,j)$ denote the minimum cost of any path from s to $(i,j)_D$, $(i,j)_I$ and $(i,j)_S$, respectively. In other words, $D(i,j)$ denotes the minimum cost of any alignment between $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$ ending with a deletion; $I(i,j)$ denotes the minimum cost of any alignment between $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$ ending with an insertion; and $S(i,j)$ denotes the minimum cost of any alignment between $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$. With proper initializations, these costs can be computed by the following recurrence relations (Gotoh, 1982; Myers and Miller, 1988):

$$D(i,j) \leftarrow \min \begin{cases} D(i-1,j) + \beta \\ S(i-1,j) + \alpha + \beta \end{cases}$$

$$I(i,j) \leftarrow \min \begin{cases} I(i,j-1) + \beta \\ S(i,j-1) + \alpha + \beta \end{cases}$$

$$S(i,j) \leftarrow \min \begin{cases} S(i-1,j-1) + \sigma(a_i,b_j) \\ D(i,j) \\ I(i,j) \end{cases}$$

An O(MN)-time, O(M + N)-space algorithm for restricted affine gap penalties

For some biosequence alignment applications, such as aligning a cDNA sequence with a genomic DNA sequence, it might be more appropriate to penalize each long gap with a constant penalty. Figure 1 depicts the process of creating eukaryotic genes (Lewin, 1994). cDNA is a single-stranded DNA complementary to an mRNA, synthesized from it by reverse transcription *in vitro*.

Assume α, β and γ are non-negative integer constants. Consider the following restricted affine gap penalties. For

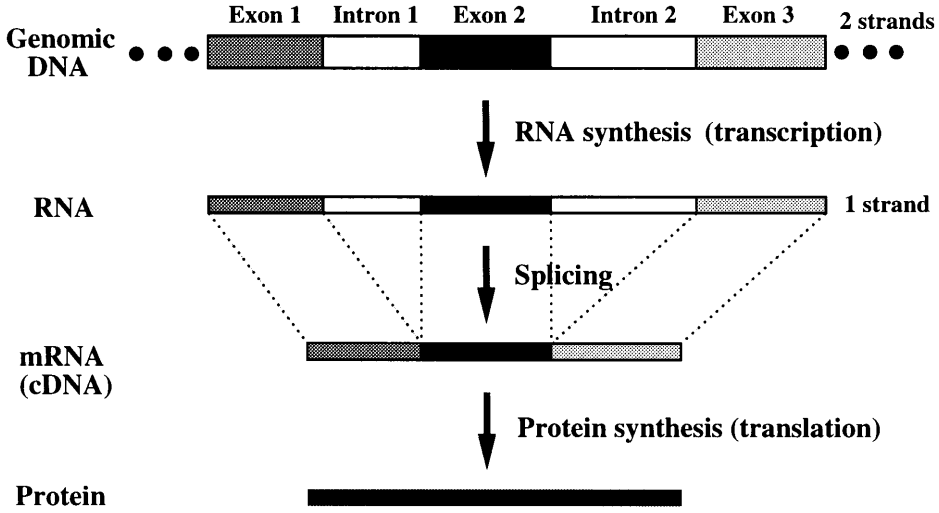

Fig. 1. Eukaryotic genes can be interrupted. Exon is any segment of an interrupted gene that is represented in the mature RNA product, while intron is removed from within the transcript by splicing together the exons on either side of it.

each $(a,b) \in \Sigma \times \Sigma$, $\sigma(a,b)$ is defined to be 0 if a is identical to b, and $\sigma(a,b)$ is γ otherwise; k-symbol gaps cost $\alpha + k\beta$, except for the situations where insertion gaps of more than l symbols are penalized by $\alpha + l\beta$, where l is a user-specified parameter. The cost of an optimal alignment of $a_1a_2 \ldots a_i$ and $b_1b_2 \ldots b_j$, denoted by $S(i,j)$, can be computed as follows:

$$S(i,j) = \min\{S(i-1,j-1) + \sigma(a_i,b_j), D(i,j), I(i,j), \bar{I}(i,j)\}$$

where:

$$D(i,j) = \min_{0 \le i' \le i-1}\{S(i',j) + \alpha + (i - i')\beta\}$$
$$I(i,j) = \min_{0 \le j' \le j-1}\{S(i,j') + \alpha + (j - j')\beta\}$$
$$\bar{I}(i,j) = \min_{0 \le j' \le j-1}\{S(i,j') + \alpha + l\beta\}$$

The following lemma simplifies the computation of $D(i,j)$.

Lemma 1. $D(i,j) = \min\{D(i-1,j) + \beta, S(i-1,j) + \alpha + \beta\}$.

Proof. By definition,

$$D(i,j) = \min_{0 \le i' \le i-1}\{S(i,j) + \alpha + (i - i')\beta\}$$
$$= \min\{\min_{0 \le i' \le i-2}\{S(i,j) + \alpha + (i - 1 - i')\beta\} + \beta, S(i-1,j) + \alpha + \beta\}$$
$$= \min\{D(i-1,j) + \beta, S(i-1,j) + \alpha + \beta\}$$

Similarly, $I(i,j) = \min\{I(i,j-1) + \beta, S(i,j-1) + \alpha + \beta\}$. To simplify the computation of $\bar{I}(i,j)$, define $I^*(i,j) = \min_{0 \le j \le j-1}\{S(i,j') + \alpha + l\beta\}$. The following lemma shows that $I^*(i,j)$ yields the same minimum cost.

Lemma 2. $\min\{I(i,j), \bar{I}(i,j)\} = \min\{I(i,j), I^*(i,j)\}$.

Proof. By definition, $I^*(i,j)$ selects the minimum over a wider range than $\bar{I}(i,j)$ does. Therefore, $\min\{I(i,j), \bar{I}(i,j)\} \ge \min\{I(i,j), I^*(i,j)\}$. However, $S(i,j) + \alpha + (j - j')\beta \le S(i,j) + \alpha + l\beta$ for $j - l < j' \le j - 1$. This implies $\min\{I(i,j), \bar{I}(i,j)\} \le \min\{I(i,j), I^*(i,j)\}$. It follows that $\min\{I(i,j), \bar{I}(i,j)\} = \min\{I(i,j), I^*(i,j)\}$.

With analogous proof techniques used in Lemma 1, the computation of $I^*(i,j)$ can be further reformulated as $\min\{I^*(i,j-1), S(i,j-1) + \alpha + l\beta\}$. In summary, with proper initializations, $S(i,j)$ can be computed by the following recurrence relationships:

$$D(i,j) \leftarrow \min\begin{cases} D(i-1,j) + \beta \\ S(i-1,j) + \alpha + \beta \end{cases}$$

$$I(i,j) \leftarrow \min\begin{cases} I(i,j-1) + \beta \\ S(i,j-1) + \alpha + \beta \end{cases}$$

$$I^*(i,j) \leftarrow \min\begin{cases} I^*(i,j-1) \\ S(i,j-1) + \alpha + l\beta \end{cases}$$

$$S(i,j) \leftarrow \min\begin{cases} S(i-1,j-1) + \sigma(a_i,b_j) \\ D(i,j) \\ I(i,j) \\ I^*(i,j) \end{cases}$$

The minimum cost $S(M,N)$ can be computed using dynamic programming techniques in $O(MN)$ time. A straightforward extension of Myers and Miller (1988) yields an $O(MN)$-time, $O(M + N)$-space algorithm for delivering an optimal alignment.

An O(NC)-time algorithm for restricted affine gap penalties

Tables D, I, I^* and S have the following important diagonal-wise monotonicity property.

Lemma 3. For $1 \le i \le M$ and $1 \le j \le N$, $D(i,j) \ge D(i-1, j-1)$, $I(i,j) \ge I(i-1,j-1)$, $I^*(i,j) \ge I^*(i-1, j-1)$ and $S(i,j) \ge S(i-1,j-1)$.

Proof. Omitted.

Hence the entries are non-decreasing along each diagonal of tables D, I, I^* and S. This property allows us to employ the 'greedy' design paradigm. For diagonal k and cost c, let $\tilde{D}(k,c)$, $\tilde{I}(k,c)$, $\tilde{I}^*(k,c)$ and $\tilde{S}(k,c)$ be the largest row i such that $D(i,k+i) = c$, $I(i,k+i) = c$, $I^*(i,k+i) = c$ and $S(i,k+i) = c$, respectively. With proper initializations, they can be computed by the following recurrence relationships:

$$\tilde{D}(k,c) \leftarrow \max\begin{cases}\tilde{D}(k+1, c-\beta)+1\\ \tilde{S}(k+1, c-\alpha-\beta)+1\end{cases}$$

$$\tilde{I}(k,c) \leftarrow \max\begin{cases}\tilde{I}(k-1, c-\beta)\\ \tilde{S}(k-1, c-\alpha-\beta)\end{cases}$$

$$\tilde{I}^*(k,c) \leftarrow \max\begin{cases}\tilde{I}(k-1, c)\\ \tilde{S}(k-1, c-\alpha-l\beta)\end{cases}$$

$$i \leftarrow \max\begin{cases}\tilde{S}(k, c-\gamma)+1\\ \tilde{D}(k,c)\\ \tilde{I}(k,c)\\ \tilde{I}^*(k,c)\end{cases}$$

$$\tilde{S}(k,c) \leftarrow \text{Jump}(i, k+1)$$

where $Jump(i,j)$ is the length of the longest common prefix of $a_{i+1}a_{i+2}\ldots a_M$ and $b_{j+1}b_{j+2}\ldots b_N$.

Fix a cost c, the tables can be computed from the lowest diagonal to the highest diagonal. Using a modification of the suffix automaton construction, it is shown that $Jump(i,j)$ can be evaluated in constant time with $O(M^2 + |\Sigma|)$ preprocessing time (Ukkonen and Wood, 1993). The algorithm terminates when $\tilde{S}(k,c) = M$. Summarizing the complexity of all stages, we state the following result.

Theorem 4. Let C be the cost of optimal alignment of A and B under restricted affine gap penalties, it can be computed in $O(NC)$ time. The preprocessing time is $O(M^2 + |\Sigma|)$.

A simple backtracking algorithm can be used to deliver the optimal alignment. For certain applications, such as aligning cDNA sequence with a genomic sequence, it is usually the case that $M << N$ and C is very small. This approach gives a very efficient solution in practice. On the other hand, for the applications where M is considerably large, the preprocessing time might turn out to be a dominant factor for running time. In this case, it is better to compute the $Jump(i,j)$ on the fly. Since the expected length of a jumped fragment by the function $Jump(i,j)$ is a small constant (e.g. for random DNA sequences, the average length is $\frac{4}{9} = \sum_{i=1}^{\infty}\frac{i}{4^i}$), the average time complexity of computing C remains $O(NC)$.

Implementation

The algorithms described in the earlier sections have been implemented as a C program, called Calign, on a Sun SPARCstation 10 running SunOS Unix. The command syntax is:

Calign file1 file2 [gap_limit [dist_limit]]

where file1 and file2 contain arbitrary sequences of characters. In our implementation, we set α and β to be 1. The parameter gap_limit specifies that the insertion gaps of more than gap_limit symbols are penalized by gap_limit + 1. The parameter dist_limit specifies the tolerable distance limit. The default values for gap_limit and dist_limit are 10 and 1000, respectively. The end gaps for the longer sequence are not penalized.

We used Calign to align the complete human genomic sequences from GenBank which have related proteins from other species (Gelfand *et al.*, 1996; Schuler *et al.*, 1996). For each genomic sequence in the test sample (see Table 2), we used a mammalian target sequence to infer its coding regions. We first extracted the coding regions of a given target sequence. Then we aligned the extracted nucleotides with the related genomic sequence. Finally, we examined whether the matching blocks of the resulting alignment correspond to the coding regions of the genomic sequence.

Table 2 summarizes the experimental results. For each case, the program reported the alignment within a second. Preliminary tests show that the alignments accurately locate almost all the coding regions. For example, in Case No. 1 of Table 2, the alignment of the human APEX genomic sequence (Seki *et al.*, 1992; Accession HUMAPEXN) and the bovine BAP 1 mRNA (Robson *et al.*, 1991; Accession BTBAP1R) shows that the matching blocks correspond to the HUMAPEXN positions 1338–1464, 1675–1862, 2429–2621 and 2752–3269, which successfully locate the coding regions of the human APEX genomic sequence. Another example (Case No. 5) is the alignment of the human *Homo sapiens* genomic sequence (Forrest *et al.*, 1991; Accession HUMCBRG) and the pig 20-beta-hydroxysteroid dehydrogenase mRNA (Tanaka *et al.*, 1992; Accession PIG20BHD). In this case, the matching blocks correspond to the HUMCBRG positions 274–566, 1112–1219 and 2608–3044, which again locate the coding regions of the human *Homo sapiens* genomic sequence. However, in Table 2, some of the coding regions are not successfully spelled out in the alignments. Most of these regions (e.g. Case No. 3) can be corrected by tuning the parameter gap limit, but if the coding region is very short (e.g. only 10 nucleotides in the human growth hormone gene of Case No. 14), then our approach might fail to report such a region.

Table 2. The experimental results. Len_g is the length of the genomic sequence; len_t is the length of the target sequence; len_c is the length of the cDNA sequence; CDS is the number of coding regions; CD is the PID of the coding regions of the target sequence; TP is true positive (number of correctly predicted coding regions); MS is the number of missed coding regions; FP is false positive (number of incorrectly predicted coding regions); DIST is the the alignment cost of the two sequences

No.	Genomic	len_g	CDS	target	len_t	CD	len_c	TP	MS	FP	DIST
1	humapexn	3730	4	btbap1r	1367	g118	957	4	0	0	135
2	humbhsd	9404	3	bt3bhsd	1632	g35	1122	3	0	0	231
3	humbnpa	1922	3	pigbnp	670	g535705	396	2	1	0	140
4	humcapg	3734	5	mmcatheg	1004	g496641	786	5	0	0	254
5	humcbrg	3326	3	pig20bhd	1230	g164294	870	3	0	0	185
6	humchymb	3279	5	dogchamc	933	g163920	750	5	0	0	170
7	humcox5b	2593	4	ratdccovb	5331	g473729	390	4	0	0	100
8	humcspa	4770	5	musccpa	1338	g309154	744	5	0	0	256
9	humfabp	5204	4	ratfabpx	564	g204088	399	4	0	0	104
10	humg0s19a	4102	3	mmscimip	1988	g297531	279	3	0	0	85
11	humg0s19b	4788	3	musstcpa	763	g533241	279	3	0	0	82
12	humgad45a	5378	4	crugad45a	4435	g409033	498	4	0	0	80
13	humgare	4754	5	pytgcbr	2321	g220647	1353	5	0	0	244
14	humghn	2657	5	bovgrowp	788	g163120	654	4	1	0	194
15	humhll4g	4428	4	ratbpgal	519	g203162	408	3	1	0	83
16	humhmg2a	4341	4	pighmg2	1153	g164492	633	4	0	0	79
17	humi309	3709	3	musstcpb	611	g533243	258	2	1	0	121
18	humibp3	10884	4	ratigfbp3a	2058	g204744	876	4	0	0	186
19	humigera	7659	5	dogigerac	1032	g303544	759	4	1	0	271
20	humil1b	7824	6	rabil1b	1377	g516633	807	6	0	0	195
21	humil4a	9900	4	ssilk	490	g1998	402	4	0	0	128
22	humil5a	3241	4	b39881	451	t:9606	451	3	1	0	256
23	humil8a	5191	4	rabnap1	1500	g16553	306	4	0	0	79
24	humil9a	4663	5	musp40m	3808	g387505	435	5	0	0	173
25	humkal2	6139	5	cfkallik	832	g414019	786	5	0	0	237
26	hummif	2167	3	musgia	600	g402717	348	3	0	0	62
27	hummis	3100	5	bovmis	2016	g163393	1728	5	0	0	424
28	humpald	7616	4	oatthyre	610	g1420	444	4	0	0	102
29	humpf4vla	1468	3	ratpf4	1675	g206091	318	3	0	0	112
30	humpgamm	834	1	mpgmut	2428	g297111	762	1	0	0	104
31	humplpspc	3409	5	mvspc	796	g1189	573	5	0	0	138
32	humpppa	2775	3	bovsmplsm	1219	g551554	288	2	1	0	59
33	humrps17	477	1	crurps17	454	g304526	408	1	0	0	36
34	humrps6b	4990	6	ratrps6	801	g206747	750	6	0	0	131
35	humsaa	3460	3	musamyaff	606	g404753	369	3	0	0	113
36	humsftp1a	4732	4	s48768	4942	g260453	747	4	0	0	205
37	humtfpb	13865	6	rabrtf	1753	g165697	879	6	0	0	259
38	humthy1a	2806	3	mthycsg	2863	g297534	486	3	0	0	129
39	humtnfba	2140	3	muslta	3294	g387407	609	3	0	0	155
40	humtnfx	3103	4	cattnfaa	705	g402366	702	4	0	0	93
41	humtpalbu	6172	6	mvegp2b	751	g505303	534	6	0	0	205
42	humtrpy1b	2609	5	dogmctrpa	995	g163983	828	5	0	0	194
43	humubilp	3583	4	musubilp	2912	g202258	474	4	0	0	101
44	humv2r	2282	3	ssvrv2a	1494	g394753	1113	3	0	0	178

It should be noted that the software package PROCRUSTES, developed by Gelfand *et al*. (1996), provides an effective approach for recognizing the coding regions. In PROCRUSTES, the spliced alignment of the genomic sequence (in nucleotides) and the target sequence (in amino acids) is used to make the prediction. Our program compares both sequences at the nucleotide level. Not only can it be used to confirm the predication made by PROCRUSTES, but it might also be used to locate possible interesting regions missed by the tool.

To demonstrate further the strength of Calign, we used it to align some UniGene clusters with the genomic sequences. For example, we aligned UniGene clusters 79058 and 115653 with genomic sequence AC004687 (175 120 nucleotides). Calign computed the alignments in just a few seconds. Interestingly, their alignments suggest an unusual genomic structure of overlapping spliced gene transcribed in reverse orientation with non-overlapping sequences. The five exons of UniGene cluster 79058 all reside in the two introns of UniGene cluster 115653 that consists of 18 overlapping EST sequences from brain and tonsil libraries. It will require further experiments to verify the overlapping genes and analyze their transcriptional regulation.

Discussion

The new program Calign is one of a suite of pairwise alignment tools that we have developed to deal with genomic DNA sequences. It can align a 100 kb sequence to a 1 megabase sequence in a few minutes on a workstation, provided that there are very few gaps between the two sequences. Future extensions include exploiting the protein homology (Salamov *et al*., 1998) or EST information (Xu *et al*., 1994), and embedding the tool as a search engine of some biological databases.

Acknowledgements

We thank Drs Webb Miller, Ueng-Cheng Yang and Jinghui Zhang for helpful conversations. We also thank the referees for improving the presentation of this paper. This work was supported in part by grant R01 LM05110 from the National Library of Medicine, National Institutes of Health, USA, and grant NSC87-2213-E-126-002 from the National Science Council, Taiwan.

References

Altschul,S., Gish,W., Miller,W., Myers,E. and Lipman,D. (1990) A basic local alignment search tool. *J. Mol. Biol*., **215**, 403–410.

Altschul,S., Madden,T.L., Schaffer,A.A., Zhang,J., Zhang,Z., Miller,W. and Lipman,D. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res*., **25**, 3389–3402.

Baeza-Yates,R.A. and Gonnet,G.H. (1994) Fast string matching with mismatches. *Inform. Comput.*, **108**, 187–199.

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

Chao,K.-M., Zhang,J., Ostell,J. and Miller,W. (1995) A local alignment tool for very long DNA sequences. *Comput. Applic. Biosci*., **11**, 147–153.

Chao,K.-M., Zhang,J., Ostell,J. and Miller,W. (1997) A tool for aligning very similar DNA sequences. *Comput. Applic. Biosci*., **13**, 75–80.

Crochemore,M., Czumaj,A., Gaasieniec,L., Jarominek,S., Lecroq,T., Plandowski,W. and Rytter,W. (1994) Speeding up two string-matching algorithms. *Algorithmica*, **12**, 247–267.

Daniels,D.L., Plunkett,G., Burland,V. and Blattner,F.R. (1992) Analysis of the *Escherichia coli* genome: DNA sequence of the region from 84.5 to 86.5 minutes. *Science*, **257**, 771–778.

Dermouche,A. (1995) A fast algorithm for string matching with mismatches. *Inform. Process. Lett*., **55**, 105–110.

Forrest,G., Akman,S., Doroshow,J., Rivera,H. and Kaplan,W. (1991) Genomic sequence and expression of a cloned human carbonyl reductase gene with daunorubicin reductase activity. *Mol. Pharmacol*., **40**, 502–507.

Gelfand,M.S., Mironov,A.A. and Pevzner,P.A. (1996) Gene recognition via spliced sequence alignment. *Proc. Natl Acad. Sci. USA*, **93**, 9061–9066.

Gotoh,O. (1982) An improved algorithm for matching biological sequences. *J. Mol. Biol*., **162**, 705–708.

Gotoh,O. (1990) Optimal sequence alignment allowing for long gaps. *Bull. Math. Biol*., **52**, 359–373.

Hardison,R.C., Chao,K.-M., Schwartz,S., Stojanovic,N., Ganetsky,M. and Miller,W. (1994) Globin Gene Server: a prototype E-mail database server featuring extensive multiple alignments and data compilation for electronic genetic analysis. *Genomics*, **21**, 344–353.

Huang,X. (1994) On global sequence alignment. *Comput. Applic. Biosci*., **10**, 227–235.

Kim,J.Y. and Shawe-Taylor,J. (1992) An approximate string-matching algorithm. *Theor. Comput. Sci*., **92**, 107–117.

Landau,G.M., Vishkin,U. and Nussinov,R. (1988) Locating alignments with k differences for nucleotide and amino acid sequences. *Comput. Applic. Biosci*., **4**, 19–24.

Lewin, B. (1994) *Genes V.* Oxford University Press, New York.

Myers,E.W. (1986) An O(ND) difference algorithm and its variations. *Algorithmica*, **1**, 251–266.

Myers,E.W. and Miller,W. (1988) Optimal alignments in linear space. *Comput. Applic. Biosci*., **4**, 11–17.

Myers,E.W. and Miller,W. (1989) Row replacement algorithms for screen editors. *ACM Trans. Program. Lang. Syst*., **11**, 33–56.

Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequences of two proteins. *J. Mol. Biol*., **48**, 443–453.

Pearson,W.R. and Lipman,D. (1988) Improved tools for biological sequence comparison. *Proc. Natl Acad. Sci. USA*, **85**, 2444–2448.

Plunkett,G., Burland,V., Daniels,D.L. and Blattner,F.R. (1993) Analysis of the *Escherichia coli* genome. III. DNA sequence of the region from 87.2 to 89.2 minutes. *Nucleic Acids Res*., **21**, 3391–3398.

Robson,C.N., Milne,A.M., Pappin,D.J. and Hickson,I.D. (1991) Isolation of cDNA clones encoding an enzyme from bovine cells that repairs oxidative DNA damage in vitro: homology with bacterial repair enzymes. *Nucleic Acids Res.*, **19**, 1087–1092.

Salamov,A.A., Nishikawa,T. and Swindells,M.B. (1998) Assessing protein coding region integrity in cDNA sequencing projects. *Bioinformatics*, **14**, 384–390.

Schuler,G.D., Epstein,J.A., Ohkawa,H. and Kans,J.A. (1996) Entrez: molecular biology database and retrieval system. *Methods Enzymol.*, **266**, 141–162.

Seki,S., Hatsushika,M., Watanabe,S., Akiyama,K., Nagao,K. and Tsutsui,K. (1992) cDNA cloning, sequencing, expression and possible domain structure of human APEX nuclease homologous to *Escherichia coli* exonuclease III. *Biochim. Biophys. Acta*, **1131**, 287–299.

Sze,S.-H. and Pevzner,P.A. (1997) Las Vegas algorithms for gene recognition: suboptimal and error-tolerant spliced alignment. *J. Comput. Biol.*, **4**, 297–309.

Sze,S.-H., Roytberg,M.A., Gelfand,M.S., Mironov,A.A., Astakhova,T.V. and Pevzner,P.A. (1998) Algorithms and software for support of gene identification experiments. *Bioinformatics*, **14**, 14–19.

Tanaka,M., Ohno,S., Nakajin,S., Shinoda,M. and Nagahama,Y. (1992) Pig testicular 20-beta-hydroxysteroid dehydrogenase exhibits carbonyl reductase-like structure and activity: cDNA cloning of pig testicular 2-beta-hydroxysteroid dehydrogenase. *J. Biol. Chem.*, **267**, 13451–13455.

Ukkonen,E. (1992) Approximate string-matching with q-grams and maximal matches. *Theor. Comput. Sci.*, **92**, 191–211.

Ukkonen,E. and Wood,D. (1993) Approximate string matching with suffix automata. *Algorithmica*, **10**, 353–364.

Waterman,M.S. (1984) Efficient sequence alignment algorithms. *J. Theor. Biol.*, **108**, 333–337.

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

Xu,Y., Mural,R. and Uberbacher,E.C. (1994) Constructing gene models from a set of accurately-predicted exons: an application of dynamic programming. *Comput. Applic. Biosci.*, **10**, 613–623.