J. Mol. Biol. (1987) **197**, 723–728

LETTERS TO THE EDITOR

A New Algorithm for Best Subsequence Alignments with Application to tRNA–rRNA Comparisons

The algorithm of Smith & Waterman for identification of maximally similar subsequences is extended to allow identification of all non-intersecting similar subsequences with similarity score at or above some preset level. The resulting alignments are found in order of score, with the highest scoring alignment first. In the case of single gaps or multiple gaps weighted linear with gap length, the algorithm is extremely efficient, taking very little time beyond that of the initial calculation of the matrix. The algorithm is applied to comparisons of tRNA–rRNA sequences from *Escherichia coli*. A statistical analysis is important for proper evaluation of the results, which differ substantially from the results of an earlier analysis of the same sequences by Bloch and colleagues.

Algorithms for locating highly similar segments (contiguous subsequences) from two different sequences have received some attention in the last few years. Unexpected relationships have been found between viral and host DNA (for examples, see Weiss, 1983; Doolittle *et al.*, 1983; Naharro *et al.*, 1984). Rapid searches of databases for these relationships are frequently made with techniques using hashing (Wilbur & Lipman, 1983). These useful approaches do not always find optimal alignments and the dynamic programming methods developed earlier are still used. In fact some analysis programs do an initial screening with hashing and then do dynamic programming for regions of possible interest. This note presents a new algorithm that produces alignments of interest by first computing the matrix of Smith & Waterman (1981) for the best alignment and then making small modifications to the matrix to produce non-intersecting subsequent alignments. The new algorithm is much simpler and more efficient than those currently in use. Readers who want a copy of our program in C should send magnetic media to us at the above address.

Dynamic programming methods of sequence comparison were introduced by Needleman & Wunsch (1970), while Sellers (1979, 1980) began work on the problem of finding well-aligned segments between two sequences. His initial work was based on distance measures and involved "forward" and "backward" matrix calculation and intersection of path graphs. Goad & Kanehisa (1982) introduced a dissimilarity measure and the concept of match density, also with forward and backward matrix calculations. Sellers (1984) shows that their criteria are equivalent to aligned segments with (1) a similarity score greater than, or equal to, some cutoff value and (2) a similarity score at least as large as scores of other intersecting alignments. Gotoh (1987) mentions that alignments of segments other than that with the largest similarity are not easily obtained with the Smith & Waterman (1981) method. Our new method easily and rapidly produces all these alignments. To avoid matrix recalculation as in Sellers' method, Boswell & McLachlin (1984) introduced exponentially damped alignment scoring. By directly locating best subsequence alignments, we avoid this device.

Take the two sequences to be $a = a_1 a_2 \ldots a_n$ and $b = b_1 b_2 \ldots b_m$. They can be either DNA or protein sequences. The similarity measure between sequence letters a and b is $s(a, b)$, where $s(a, b) > 0$ if $a = b$ and $s(a, b) < 0$ for at least some cases of $a \neq b$. Insertions or deletions of length k receive weight $-w_k$. The observation of Smith & Waterman (1981) is that negative scoring alignments are of no interest. $S(a, b)$ is defined to be the best (largest) score from aligning a and b. Define:

$$H_{i,j} = \max\{0; S(a_x a_{x+1} \ldots a_i, b_y b_{y+1} \ldots b_j): 1 \leqq x \leqq i, 1 \leqq y \leqq j\}. \quad (1)$$

$H_{i,j}$ is the best score of any alignment ending at a_i and b_j or 0, whichever is larger. The similarity algorithm is started with:

$$H_{i,0} = H_{0,j} = 0,\ 1 \leqq i \leqq n,\ 1 \leqq j \leqq m.$$

Then:

$$H_{i,j} = \max\{0, H_{i-1,j-1} + s(a_i, b_j), E_{i,j}, F_{i,j}\}, \quad (2)$$

where:

$$E_{i,j} = \max_{1 \leqq k \leqq j}\{H_{i,j-k} - w(k)\}, \quad (3)$$

$$F_{i,j} = \max_{1 \leqq k \leqq i}\{H_{i-k,j} - w(k)\}. \quad (4)$$

When $w_k = u + vk$, Gotoh (1982) has shown that this $O(n^3)$ algorithm can be reduced to $O(n^2)$. Waterman (1984), obtained a reduction for concave w_k related to that obtained for linear w_k. For algorithms where insertions and deletions are obtained by inserting or deleting one letter at a time, the algorithms are already $O(n^2)$. This is the case with the algorithms of Sellers (1979, 1980) and Goad & Kanehisa (1982), for example.

The original Smith & Waterman algorithm found alignments by tracing back from (i,j) where

0022-2836/87/200723-06 $03.00/0

$H_{i,j} = \max_{k,l} H_{k,l}$. The optimal segment alignments must end at (i,j) achieving the maximum. Sometimes alignments can have 0 score segments added on to them. We simply do not allow these, as is described below. It is in fact possible for related sequences to have many optimal alignments. For example, the rRNA sequences 5 S *Escherichia coli* and 5 S *Mycoplasma capricolum*, when compared with $s(a,b) = +1$ if $a = b$ and -1 if $a \neq b$, and $w_1 = -2$, have similarity 22, and there are 52,020 difficulties, our implementation traces back one optimal alignment from (i,j) instead of all optimal alignment from (i,j) instead of all optimal alignments. Other scientists may wish to have all optimal alignments; it is easy to modify the algorithm given here to do that. We now describe how we choose our optimal alignment.

If there are ties for $\max_{i,j} H_{i,j}$, we choose among the possible positions to begin the traceback as follows:

$$\text{If } H_{i,j} = H_{k,l} \text{ and } i+j < k+l, \quad \text{traceback from } (i,j). \tag{5}$$

$$\text{If } H_{i,j} = H_{k,l},\ i+j = k+l, \text{ and } i < k, \quad \text{traceback from } (i,j). \tag{6}$$

Equation (5) is to eliminate 0 score segments at the end of an alignment, while equation (6) is to assure that similar alignments are output when the sequences are reversed. (By reversed we mean that the sequences are written backwards.) Equation (6) simply gives a rule to determine completely the (i,j) value at which to begin the traceback. Next it is necessary to choose among the (p,q) values where the traceback ends. To be consistent with equation (5), if (p,q) and (r,s) both end tracebacks from (i,j) we choose (p,q) if $r+s < p+q$. To be consistent with equation (6), if $p+q = r+s$, we choose (p,q) if $p > r$. The result of these rules is that, for both the sequences reversed and unreversed, the alignment begins and ends at the same locations. In between we always take diagonal steps (match or mismatch) whenever possible, but the alignments can differ, usually in minor ways, if at all, when sequences are reversed.

Having refined the best alignment definition slightly, we turn to finding the other alignments of interest. Two alignments are said to intersect if they have a match or mismatch in common. It is possible to extend this definition to require a larger number or even a percentage of matches and/or mismatches in common, but we have found the definition given here to be very useful. This definition can also be extended to include insertions and deletions. We want to find the next largest scoring alignment that does not intersect those already output. The simplest idea is to recompute the matrix, simply not allowing matches or mismatches that were involved in alignments already output. (Kruskal & Sankoff (1983) made a related proposal in which they recommended recomputation forcing any (i,j) involved in an output alignment to have the value 0.) Of course these procedures involve more computation. Each additional alignment requires on the average $n^2/4$ matrix entries to be recomputed, since only elements below and to the right of (i,j) can be affected by changing $H_{i,j}$. (We take (0,0) to be in the upper lefthand corner of the matrix.) To obtain $1+A$ alignments takes $(1+A/4)$ times as long as obtaining the first or best. While this most general version of our method is somewhat costly in terms of time, it is extremely simple to implement; the intersecting path graph methods are not. We now turn to reducing greatly the time requirements of the algorithm. The reduction is based on the observation that only matrix elements "near" the output alignment change on recomputation; the vast majority of the recomputed matrix H^* is identical with the original matrix H (see Fig. 1 for an example of H and H^*).

Suppose we have an algorithm with $w_k = vk$:

$$H_{i,j} = \max\{0,\ H_{i-1,j-1} + s(a_i, b_j), \quad H_{i-1,j} - v,\ H_{i,j-1} - v\}. \tag{7}$$

This algorithm considers one-letter insertions and deletions at a time and is simpler than the general linear w_k algorithm given in equations (2) to (4). As mentioned above, only matrix elements below and to the right of alignment entries can differ from the original matrix values. Take the upper leftmost alignment position (i,j). The computations start with:

$$H^*_{i,j} = \max\{0, H_{i-1,j} - v, H_{i,j-1} - v\}.$$

Notice that the match with which the alignment ended is not allowed. Consider the column (k,j), where $i < k$. Recompute each entry (k,j), $k = i+1, i+2, \ldots$ until the new value $H^*_{k,j}$ equals the previous $H_{k,j}$. Then since $H^*_{l,j-1} = H_{l,j-1}$, for all l, it is clear that $H^*_{l,j} = H_{l,j}$ for all $k < l$. A similar calculation for the row (i,k), $j < k$, allows us to stop whenever the new $H^*_{i,k}$ equals $H_{i,k}$. We proceed by induction. Notice that in later calculations we must go to at least the position that was necessary for the preceding row or column. This greatly reduces the recalculation necessary. If the output alignments have length L and we output A alignments, the total computation required is approximately $O(n^2) + AO(L^2)$. If $n = 1000$, $A = 10$, and $L = 20$, then the complexity is $10^6 + 10 \times 20^2 = 10^6 + 4 \times 10^3$. Notice that the alignments are produced with much less computation than the original matrix. $A = 2500$ alignments of length 20 must be processed to have the alignment calculations balance that of the original matrix calculation.

Next we turn to obtaining similar efficiencies for the linear gap functions, $w(k) = u + vk$, where k is gap length. Gotoh (1982) showed that the time for the multiple gap algorithm of Waterman *et al.* (1976) could be reduced to $O(n^2)$ for linear gap functions by altering the last two recursions, equations (3) and (4), to:

$$E_{i,j} = \max\{H_{i,j-l} - (u+v), E_{i,j-l} - v\},$$
$$F_{i,j} = \max\{H_{i-l,j} - (u+v), F_{i-l,j} - v\}.$$

(a)

	A	G	T	C	C	G	A	G	G	G	C	T	A	C	T	C	T	A	C	T	G	A	A	C
C	0	0	0	10	10	0	0	0	0	0	10	0	0	10	0	10	0	0	10	0	0	0	0	10
C	0	0	0	10	20	1	0	0	0	0	10	1	0	10	1	10	1	0	10	1	0	0	0	10
A	10	0	0	0	1	11	11	0	0	0	0	1	11	0	1	0	1	11	0	1	0	10	10	0
A	10	1	0	0	0	0	21	2	0	0	0	0	11	2	0	0	0	11	2	0	0	10	20	1
T	0	1	11	0	0	0	1	12	0	0	0	10	0	2	12	0	10	0	2	12	0	0	1	11
C	0	0	0	21	10	0	0	0	3	0	10	0	1	10	0	22	2	1	10	0	3	0	0	11
T	0	0	10	1	12	1	0	0	0	0	0	20	0	0	20	2	32	12	0	20	0	0	0	0
A	10	0	0	1	0	3	11	0	0	0	0	0	30	10	0	11	12	42	22	2	11	10	10	0
C	0	1	0	10	11	0	0	2	0	0	10	0	10	40	20	10	2	22	52	32	12	2	1	20
T	0	0	11	0	1	2	0	0	0	0	0	20	0	20	50	30	20	2	32	62	42	22	2	0
A	10	0	0	2	0	0	12	0	0	0	0	0	30	10	30	41	21	30	12	42	53	52	32	12
C	0	1	0	10	12	0	0	3	0	0	10	0	10	40	20	40	32	12	40	22	33	44	43	42
T	0	0	11	0	1	3	0	0	0	0	0	20	0	20	50	30	50	30	20	50	30	24	35	34
G	0	10	0	2	0	11	0	10	10	10	0	0	11	0	30	41	30	41	21	30	60	40	20	26
C	0	0	1	10	12	0	2	0	1	1	20	0	0	21	10	40	32	21	51	31	40	51	31	30
T	0	0	10	0	1	3	0	0	0	0	0	30	10	1	31	20	50	30	31	61	41	31	42	22
T	0	0	10	1	0	0	0	0	0	0	0	10	21	1	11	22	30	41	21	41	52	32	22	33
G	0	10	0	1	0	10	0	10	10	10	0	0	1	12	0	2	13	21	32	21	51	43	23	13
C	0	0	1	10	11	0	1	0	1	1	20	0	0	11	3	10	0	4	31	23	31	42	34	33
A	10	0	0	0	1	2	10	0	0	0	0	11	10	0	2	0	1	10	11	22	14	41	52	32
G	0	20	0	0	0	11	0	20	10	10	0	0	2	1	0	0	0	0	1	2	32	21	32	43
T	0	0	30	10	0	0	2	0	11	1	1	10	0	0	11	0	10	0	0	11	12	23	12	23
A	10	0	10	21	1	0	10	0	0	2	0	0	20	0	0	2	0	20	0	0	2	22	33	13
C	0	1	0	20	31	11	0	1	0	0	12	0	0	30	10	10	0	0	30	10	0	2	13	43

(b)

	A	G	T	C	C	G	A	G	G	G	C	T	A	C	T	C	T	A	C	T	G	A	A	C
C	0	0	0	10	10	0	0	0	0	0	0*	0*	0	10	0	10	0	0	10	0	0	0	0	10
C	0	0	0	10	20	1	0	0	0	0	10*	0*	0*10		1	10	1	0	10	1	0	0	0	10
A	10	0	0	0	1	11	11	0	0	0	0	1*	0*	0*	1	0	1	11	0	1	0	10	10	0
A	10	1	0	0	0	0	21	2	0	0	0	0	11*	0*	0*	0	0	11	2	0	0	10	20	1
T	0	1	11	0	0	0	1	12	0	0	0	10	0	2*	0*	0*10		0	2	12	0	0	1	11
C	0	0	0	21	10	0	0	0	3	0	10	0	1	10	0*	0*	0*	1*10		0	3	0	0	11
T	0	0	10	1	12	1	0	0	0	0	0	20	0	0	20	0*	0*	0*	0*20		0	0	0	0
A	10	0	0	1	0	3	11	0	0	0	0	0	30	10	0	11*	0*	0*	0*	0*11	*10	10	0	
C	0	1	0	10	11	0	0	2	0	0	10	0	10	40	20	10	2*	0*	0*	0*	0*	2*	1	20
T	0	0	11	0	1	2	0	0	0	0	0	20	0	20	50	30	20		0*	0*	0*	0*	0*	0* 0
A	10	0	0	2	0	0	12	0	0	0	0	0	30	10	30	41	21	30	*10	0*	0*	0*	10*	0* 0
C	0	1	0	10	12	0	0	3	0	0	10	0	10	40	20	40	32	12	40	*20	0*	0*	1*	20
T	0	0	11	0	1	3	0	0	0	0	0	20	0	20	50	30	50	30	20	50	*30	*10	0*	0* 0
G	0	10	0	2	0	11	0	10	10	10	0	0	11	0	30	41	30	41	21	30	60	40	*20	0* 0
C	0	0	1	10	12	0	2	0	1	1	20	0	0	21	10	40	32	21	51	31	40	51	31	30
T	0	0	10	0	1	3	0	0	0	0	0	30	10	1	31	20	50	30	31	61	41	31	42	22
T	0	0	10	1	0	0	0	0	0	0	0	10	21	1	11	22	30	41	21	41	52	32	22	33
G	0	10	0	1	0	10	0	10	10	10	0	0	1	12	0	2	13	21	32	21	51	43	23	13
C	0	0	1	10	11	0	1	0	1	1	20	0	0	11	3	10	0	4	31	23	31	42	34	33
A	10	0	0	0	1	2	10	0	0	0	0	11	10	0	2	0	1	10	11	22	14	41	52	32
G	0	20	0	0	0	11	0	20	10	10	0	0	2	1	0	0	0	0	1	2	32	21	32	43
T	0	0	30	10	0	0	2	0	11	1	1	10	0	0	11	0	10	0	0	11	12	23	12	23
A	10	0	10	21	1	0	10	0	0	2	0	0	20	0	0	2	0	20	0	0	2	22	33	13
C	0	1	0	20	31	11	0	1	0	0	12	0	0	30	10	10	0	0	30	10	0	2	13	43

Figure 1. The maximum segments matrix H for computing a = CCAATCTACTACTGCTTGCAGTAC with b = AGTCCGAGGGCTACTCTACTGAAC appears in (a) while H^* appears in (b). The entries H^* that required recalculation are indicated by an asterisk on the right. Paths for the alignments can be followed by tracing the boxed entries.

To implement our algorithm for these recursions, the same procedure is followed as for the single gap case, except that here we stop along a row or column when all three quantities $H_{i,j}$, $E_{i,j}$ and $F_{i,j}$ are equal to $H^*_{i,j}$, $E^*_{i,j}$ and $F^*_{i,j}$, respectively. Therefore for general linear gap functions we have achieved the same efficiencies as for the single gap case.

In Figure 1, the sequence a = CCAATCTACTAC-TGCTTGCAGTAC is compared to b = AGTCCG-AGGGCTACTCTACTGAAC with $s(a,a) = 1$, $s(a,b) = -0{\cdot}9$ if $a \neq b$, and deletions of length k have weight $w_k = -2k$. Figure 1(a) shows the matrix with entries multiplied by 10 (so that integer arithmetic can be used). The maximum entry has a score of 6·2, associated with the

alignment:

CCAATCTACT
CTACTCTACT.

The matrix is processed again to produce H^*. The entries where recomputation has been performed are marked on the right by an asterisk. Only 63 of the 576 entries were recomputed. The maximum entry in H^* is 6·1 with the resulting alignment:

CTACTACTGCT
CTACT-CTACT.

Much attention has been given to comparisons of rRNA sequence and structure across a diverse set of organisms. The powerful concept of the conservation of structure has been used to derive the secondary structure of 5 S rRNA (Fox & Woese, 1975; Nishikawa & Takemura, 1974); 16 S rRNA (Noller & Woese, 1981) and of 23 S rRNA (Noller *et al.*, 1980). These studies are based on conservation of sequence and structure over evolutionary time. There are also studies that consider the possibility of ancestral RNA sequences from which other RNA sequences, if not all nucleic acid sequences, have evolved. Eigen & Winkler-Oswatitsch (1981) argue that tRNA was an early self-replicating gene. Bloch *et al.* (1983) carry the arguments further and suggest that matchings between tRNA and rRNA sequences are evidence of common evolutionary origins and are therefore homologies. Any such argument should be based on good computer search algorithms and on careful statistical considerations. We illustrate the power of our new algorighm with a study of tRNA–rRNA comparisons of *E. coli* sequences. It is necessary to evaluate the comparisons statistically.

The parameters used for the algorithm remain 1 for a match, −0·9 for a mismatch, and $w_k = -2k$ for insertions or deletions. The source of the sequences is GenBank and we compare 33 tRNA sequences with 16 S rRNA, all from *E. coli*. The results are shown in Table 1. Before we discuss the results, it is necessary to describe the method used to obtain statistical significance.

Recently the statistical nature of matchings between random sequences has been studied and a new probability distribution derived. This work began with the empirical study of Smith *et al.* (1985) and has been continued by Arratia *et al.* (1986) and by Waterman *et al.* (1987). The mathematical results, from the last paper cited, state that for a wide range of penalties, H is proportional to the logarithm of the product of the lengths of the sequences being compared. The sequence lengths should not be too disparate. The variance is constant and does not depend on the sequence lengths. In the paper by Smith *et al.* (1985), over 20,000 scores from sequence comparisons of unrelated DNA sequences of a great many organisms, with the same algorithm parameters used above, gave the following fit for the average score A,

$$A = 2{\cdot}55 \log(nm) - 8{\cdot}99,$$

Table 1

The summary of comparisons of tRNA and 16 S sequences from E. coli

E. coli tRNA	Position in tRNA	Position in 16 S	Score	#s	Reference: Sprinzl & Gauss (1982)	Reference: GenBank accession number
Ala-1a	50	591	12·2	−0·2	0010	K00139
Ala-1b	50	591	12·2	−0·1	0011	K00140
Cys	74	1328	21·0	6·2	0410	K00179
Asp-1	56	1424	10·8	−1·1	0310	K00169
Glu-1	73	280	10·9	−0·8	0610	K00188
Glu-2	73	280	12·8	0·6	0620	K00189
Phe	47	685	13·0	0·6	1410	K01552
Gly-1	62	797	9·4	−1·4	0710,0711	K00196
Gly-2	54	590	9·5	−1·2	0720,0721	K00198
Gly-3	62	295	14·4	1·5	0730	K01549
His-1	65	1367	13·2	1·1	0810	K00211
Ile-1	68	967	13·6	0·9	0910	K00217
Ile-2	66	1317	14·0	1·3	0911	K00218
Lys	34	1090	10·7	−0·5	1110	K00282
Leu-1	87	1412	13·8	0·7	1010	K01550
Leu-2	72	385	11·7	−0·7	1011	K01551
Leu-5	59	812	13·4	0·4	1012	K00225
Met-f	76	1112	12·0	−0·3	1310	K00305
Met-m	75	807	11·4	−0·2	1210	K00296
Asn	52	480	15·3	2·4	0210	K00164
Gln-1	32	956	11·8	0·1	0510	K00181
Gln-2	39	961	12·1	0·2	0520	K00182
Arg-1	76	355	13·3	0·7	0110	K00152
Arg-2	39	728	12·8	0·3	0111	K00153
Ser-1	45	135	11·1	−1·3	1610	K01555
Ser-3	69	347	13·8	0·3	1620	K01556
Thr-ggt	18	31	10·1	−1·3	1710	K00275
Val-1	69	1541	11·9	−0·2	2010	K01559
Val-2a	34	1540	11·3	−0·7	2020	K01560
Val-2b	34	1540	11·3	−0·4	2021	K01561
Trp	37	728	11·0	−0·7	1810,1811,1814	K00260
Tyr-1	56	484	11·7	−0·4	1910,1911	K00266
Tyr-2	69	1188	10·9	−0·9	1910	K00267

(Position in tRNA, Position in 16 S) = (i,j); Score = $\max_{i,j} H_{i,j}$; #s = (score − average)/s, where s is estimated standard deviation.

where the base of the logarithm is $1/p$, p = the probability of random bases from the sequences matching, and n and m are sequence lengths. The standard deviation is $s = 1{\cdot}55$. This distribution is not normal, and the normal distribution will lead to the assignment of significance where unwarranted. This approach is very useful in screening out matches that might otherwise seem significant.

The above formula could not be used directly in our examples because the length of the tRNAs varies from 74 to 88 in our case while 16 S is of length 1542. A simulation performing 100 comparisons of random sequence pairs for each $n = 74, 76, \ldots, 88$ was made and the following fit resulted for the average score A:

$$A = 5{\cdot}04 \log(1542n) - 30{\cdot}95,$$

and the estimate of standard deviation is $s = 1{\cdot}49$. In each comparison of Table 1 the matching probability was calculated and the average score was found by the above formula. From this #s = (score − average)/s was found. Clearly at most two

of the comparisons warrant attention. In the case of a normal distribution there is about 2·5% probability of being more than two units of standard deviation above the average; in the distribution here there is about 4% likelihood of being more than two units of standard deviation above the average. So, in about 20 independent comparisons, we expect about one to be 2*s* above the average.

The most significant match is that between cysteine tRNA and 16 S. It has a score of 21·0 and has 40 matches, ten mismatches, and five bases in insertion/deletions. The alignment is:

5′ AGCGGA--TTGCAAA-TCCGTC-TAGTCCGG-TTCGACTCCGGAACGCGCCTCCA tRNA
5′ AGCGGACCTCATAAAGTGCGTCGTAGTCCGGATTGGAGTCTGCAACTCGACTCCA 16 S

To evaluate this alignment we did two simulations of size 1000, where random sequences with the same base frequencies and lengths were generated and the similarity score calculated. The score 21·0 has an approximate significance of 10^{-3}. When we find the second best matching between these two sequences, we obtain a score of 11·7, with a match ending at 62 in tRNA and 721 in 16 S:

5′ TGTAGCGGATTGCAAATCCGTCTAG-TCCGGTTCGACT-CCGG tRNA
5′ TGTAGCGG--TG-AAATGCGTAGAGATCTGG-AGGAATACCGG 16 S

This second alignment has a score slightly above the average for random sequences of the same length; its more interesting feature is that it is essentially contained in the region of Cys tRNA given in the first alignment. If the second alignment is assumed to be independent of the first, we can roughly estimate significance as follows: say the score would be as large as it is with probability 1/2. The second alignment of 40 bases can be located along the tRNA in about 40 ways, while it can be located within the region of first alignment in about ten ways, giving a probability of 1/4 of the alignment overlapping in the observed fashion. The product of these numbers is 1/8, probably not as small as might have been expected. Still the likelihood of finding both these alignments in random sequences is now estimated to be about 10^{-4}.

The same comparisons were done for the tRNAs with 16 S reversed, 5 S, 5 S reversed, 23 S and 23 S reversed, as well as all the reversed sequences complemented. The only statistically significant alignment obtained was between isoleucine tRNA and 5 S reversed. The score was 15, which is significant at about 0·003. The alignment, ending at

Table 2

The results of a simulation of ten pairs of random sequences, one sequence in each pair with length and base composition that of Ala-tRNA and the second of 16 S rRNA

Alignment	Score	Matches	Mismatches	Indels
CTACTGGGG-AACC CTAATGGGGCAACC	8·2	12	1	1
TCGAAGTGACTCGTCGTGTGTGCAT TCGAAGT-AC-GGTAGAGTGAGCAT	8·6	19	4	2
GTAATGGCTGCCTGGGCACGCCTCA GTTATGGTTGCC-GGGGGTGCCTCA	9·6	19	5	1
ACGCTCGTACACCG ACGCTC-TTCACCG	8·2	12	1	1
AAGCGGCGCGGT-GAGCCA AAGCGGCGTGATCGACCCA	8·4	15	3	1
GGTAGATAT GGTAGATAT	9·0	9	0	0
CGAGTCCT CGAGTCCT	8·0	8	0	0
GACGCACCGTC-CATCT GACCCACCGGCGCATCT	8·8	14	2	1
CTCCGACGGCTTGTGTGGGG CTCCG-CCACTTGTGTGGGG	11·8	17	2	1
GGATG-ACCCAC GGATGTACCCAC	8·6	11	0	1

The algorithm parameters were 1 for matches, −1·4 for mismatches, and −2·4 for insertion/deletions. The best subsequence alignment for each pair of sequences is given.

position 72 in 5 S and 54 in Ile-tRNA, is:

5′ TCAGGTGGTTAGAGCGCACCCCTGATAAGGGTGAGGTCGGTGG Ile-tRNA
3′ TCAAGGGATGAGAGCGTACCCCT-CT-GGGGTGTGAT-GGTAG 5 S rev

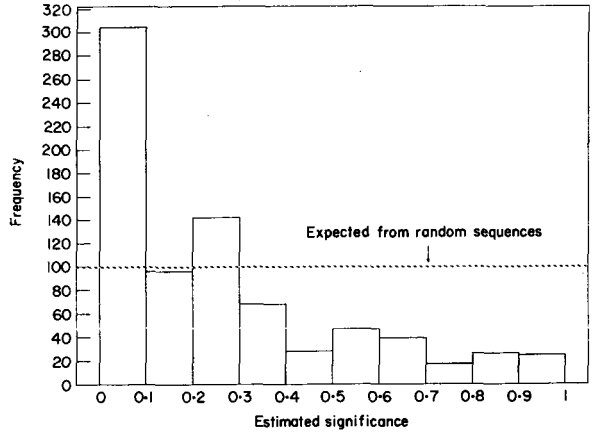

Figure 2. The results of a simulation comparing 1000 pairs of sequences, one sequence in each pair of length and base composition is that of Ala-tRNA and the second is that of 16 S rRNA. The algorithm parameters were 1 for matches, −1 for mismatches, and −2 for insertion/deletions. The significance was estimated by the method of Bloch *et al.* (1983). These results should be compared with a horizontal line at height 100, which is expected for significance estimates from random sequences. While no significance larger than 1·0 should appear, of the 100 comparisons, 208 were in this range: 108 were larger than 2·5; 49 larger than 5·0; 23 larger than 10·0. The largest value was 76·7.

The above analysis finds two statistically significant matches between *E. coli* tRNA and rRNA. In sharp contrast, the work of Bloch *et al.* (1983) finds so many significant matches that the authors conclude that "matches are too frequent and extensive to be attributed to coincidence". The difference between the results lies in the methods of finding matches and assigning significance (Goad & Kanehisa, 1982). We estimate the significance of the Cys tRNA–16 S match to be approximately 10^{-3} while Bloch *et al.* estimate it to be 3×10^{-5}. In simulations, we find that their method assigns significance of less than 10% to over 30% of the pairs of random sequences, with the significances generally being biased to small values (see Fig. 2). In Table 2, we show the best segment alignments from ten pairs of simulated sequences to illustrate how impressive the results of aligning random sequences can be.

This work was supported by grants from the National Institutes of Health and the System Development Foundation.

Michael S. Waterman
Mark Eggert
Departments of Mathematics and Molecular Biology
University of Southern California
Los Angeles, CA 90089-1113, U.S.A.

Received 5 August 1986 and in revised form
16 June 1987

References

Arratia, R., Gordon, L. & Waterman, M. S. (1986). *Ann. Stat.* **14**, 971–993.
Bloch, D. P., McArthur, B., Widdowson, R., Spector, D., Guimaraes, R. C. & Smith, J. (1983). *J. Mol. Evol.* **19**, 420–428.
Boswell, D. R. & McLachlin, A. D. (1984). *Nucl. Acids Res.* **12**, 457–464.
Doolittle, R. F., Hunkapiller, M. W., Hood, L. E., Devare, S. G., Robbins, K. C., Aaronson, S. A. & Antoniades, H. M. (1983). *Science,* **221**, 275–276.
Eigen, M. & Winkler-Oswatitsch, R. (1981). *Naturwissenschaften,* **68**, 282–292.
Fox, G. E. & Woese, C. R. (1975). *Nature (London),* **256**, 505–507.
Goad, W. B. & Kanehisa, M. I. (1982). *Nucl. Acids Res.* **10**, 247–263.
Gotoh, O. (1982). *J. Mol. Biol.* **162**, 705–708.
Gotoh, O. (1987). *CABIOS* **3**, 17–20.
Kruskal, J. B. & Sankoff, D. (1983). In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison* (Kruskal, J. B., and Sankoff, D., eds), pp. 265–310, Addison-Wesley, London.
Naharro, G., Robbins, K. C. & Reddy, E. P. (1964). *Science,* **223**, 63–66.
Needleman, S. B. & Wunsch, C. D. (1970). *J. Mol. Biol.* **48**, 443–453.
Nishikawa, K. & Takemura, S. (1974). *J. Biochem.* **76**, 935–947.
Noller, H. F. & Woese, C. (1981). *Science,* **212**, 403–411.
Noller, H. F., Kop, J., Wheaton, V., Brosius, J., Gutell, R. R., Kopylov, A. M., Dohme, F., Herr, W., Stahl, D. A., Gupta, R. & Woese, C. (1980). *Nucl. Acids Res.* **9**, 6167–6189.
Sellers, P. H. (1979). *Proc. Nat. Acad. Sci., U.S.A.* **76**, 3041.
Sellers, P. H. (1980). *J. Algorith.* **1**, 359–373.
Sellers, P. H. (1984). *Bull. Math. Biol.* **46**, 501–514.
Smith, T. F. & Waterman, M. S. (1981). *J. Mol. Biol.* **147**, 195–197.
Smith, T. F., Waterman, M. S. & Burks, C. (1985). *Nucl. Acids Res.* **13**, 645–656.
Sprinzl, M. & Gauss, D. H. (1982). *Nucl. Acids Res.* **10**, 1–55.
Waterman, M. S. (1984). *J. Theoret. Biol.* **108**, 333–337.
Waterman, M. S., Smith, T. F. & Beyer, W. A. (1976). *Advan. Math.* **20**, 367–387.
Waterman, M. S., Gordon, L. & Arratia, R. (1987). *Proc. Nat. Acad. Sci., U.S.A.* **84**, 1239–1243.
Wilbur, W. J. and Lipman, D. J. (1983). *Proc. Nat. Acad. Sci., U.S.A.* **80**, 726–730.
Weiss, T. (1983). *Nature (London),* **304**, 12.

Edited by S. Brenner