Bulletin of Mathematical Biology Vol. 54, No. 4, pp. 521–536, 1992.
Printed in Great Britain.

0092-8240/92$5.00+0.00
Pergamon Press plc

A LOCAL ALGORITHM FOR DNA SEQUENCE ALIGNMENT WITH INVERSIONS

■ MICHAEL SCHÖNIGER and MICHAEL S. WATERMAN
Departments of Mathematics and Molecular Biology,
University of Southern California,
Los Angeles, CA 90089-1113, U.S.A.

A dynamic programming algorithm to find all optimal alignments of DNA subsequences is described. The alignments use not only substitutions, insertions and deletions of nucleotides but also inversions (reversed complements) of substrings of the sequences. The inversion alignments themselves contain substitutions, insertions and deletions of nucleotides. We study the problem of alignment with non-intersecting inversions. To provide a computationally efficient algorithm we restrict candidate inversions to the K highest scoring inversions. An algorithm to find the J best non-intersecting alignments with inversions is also described. The new algorithm is applied to the regions of mitochondrial DNA of *Drosophila yakuba* and mouse coding for URF6 and cytochrome b and the inversion of the URF6 gene is found. The open problem of intersecting inversions is discussed.

1. Introduction. DNA sequence data continue to have a profound effect on biology. The information content of genes is revealed in DNA sequence and, consequently, inferences can be made about the relatedness of genes or more generally about segments of DNA sequences. The sequences under study can be found in different species such as human, chimpanzee and gorilla where a question is to determine the closest relative to humans. The sequences can be of the same gene chosen from members of a population, such as the ADH gene of *Drosophila*. Finally the sequences can be related genes from a single organism, such as α, β and δ hemoglobin genes. In examples such as we have given the sequences must be compared and similar or homologous positions identified. Due to the large data set (over 50×10^6 base pairs of sequence are in DDBJ, EMBL and GenBank as of spring 1991) and the combinatorial nature of sequence alignment, DNA and protein sequences are usually compared and aligned using an algorithm implemented on a computer.

Dynamic programming algorithms are a method of choice for sequence comparison. Other methods such as FASTA are used in database searches because of speed and even these methods use dynamic programming at one stage (Pearson and Lipman, 1988). The standard dynamic programming sequence alignment algorithms produce alignments that maximize a specific objective function. The objective function or score is the sum of weighted matches and mismatches, usually with negative weights for insertions and deletions. The resulting alignments give explicit relationships between the sequences; the substitutions, insertions and deletions required to transform one sequence into another. See Waterman (1984, 1989) for a general discussion of these topics.

While dynamic programming is often used for sequence comparison, there are some drawbacks. One is the limitation of the evolutionary transformations to substitutions, insertions and deletions. Duplications and inversions are common events in molecular evolution (Howe *et al.*, 1988; Zhou *et al.*, 1988). Although Wagner (1983) has studied sequence comparison with inversions of adjacent letters, his results are negative since including inversions is very costly in terms of computer time. Also his inversions are transpositions of adjacent letters, transformations of much less interest to us than inversions of longer segments of DNA. Therefore including inversions has seemed computationally impractical, and there has been no other rigorous work since Wagner. In this paper we take a different approach and give a practical solution to sequence comparison with non-overlapping inversions.

2. Dynamic Programming. Needleman and Wunsch (1970) introduced the first dynamic programming algorithm for sequence comparison. It was recast in its present form in Smith *et al.* (1981) where similarity and distance algorithms were shown to be equivalent or duals of one another. Here we study similarity algorithms. For explicitness let $\mathbf{a} = a_1 a_2 \ldots a_n$ and $\mathbf{b} = b_1 b_2 \ldots b_m$ be two DNA sequences. If a and b are letters of the individual sequences, $s(a, b)$ is a real valued function. Gaps are inserted by insertion of "–". Gaps of k contiguous letters are given negative weight $w(k)$ and were introduced by Waterman *et al.* (1976). In the case $w(k) = kw(1)$ the objective function is easy to write out. Insert "–" in $\mathbf{a}$ and in $\mathbf{b}$ so that the new sequences $\mathbf{a}^*$ and $\mathbf{b}^*$ are of the same (finite) length. The alignment:

$$\begin{array}{c} a_1^* \, a_2^* \ldots a_L^* \\ b_1^* \, b_2^* \ldots b_L^* \end{array}$$

should not have two aligned "–"s. Thus $\max\{n, m\} \leqslant L \leqslant n+m$. Set:

$$A(\mathbf{a}^*, \mathbf{b}^*) = \sum_{i \geqslant 1} s(a_i^*, b_i^*)$$

where $s(-, b) = s(a, -) = w(1)$. Now the score $S(\mathbf{a}, \mathbf{b})$ for best alignment of $\mathbf{a}$ with $\mathbf{b}$ is defined by:

$$S(\mathbf{a}, \mathbf{b}) = \max_{\mathscr{A}} A(\mathbf{a}^*, \mathbf{b}^*)$$

where $\mathscr{A}$ is the set of all alignments. If $w(k) \neq kw(1)$, some modifications must be made to the definition.

The beauty of dynamic programming is that it provides a simple recursion to compute $S(\mathbf{a}, \mathbf{b})$. Set:

$$S(i,j)=S(a_1a_2 \ldots a_i, b_1b_2, \ldots b_j)$$

where $S(0,0)=0$, $S(0,j)=S(-, b_1 \ldots b_j)=w(j)$ and $S(i,0)=w(i)$. Then:

$$S(i,j)=\max\left\{S(i-1,j-1)+s(a_i,b_j),\ \max_{1\leqslant k\leqslant i}\{S(i-k,j)+w(k)\},\right.$$
$$\left.\max_{1\leqslant l\leqslant j}\{S(i,j-l)+w(l)\}\right\}. \qquad (1)$$

Of course $S(n,m)=S(\mathbf{a},\mathbf{b})$. This algorithm takes time $O(n^2m+m^2n)$ or $O(n^3)$ if $n=m$. In the case of a linear weight function $w(k)=\alpha+\beta k$, there is a nice reduction of computing time to $O(n^2)$ due to Gotoh (1982):

$$E(i,j)=\max\{E(i-1,j)+\beta,\ S(i-1,j)+\alpha+\beta\}$$
$$F(i,j)=\max\{F(i,j-1)+\beta,\ S(i,j-1)+\alpha+\beta\}$$

and

$$S(i,j)=\max\{S(i-1,j-1)+s(a_i,b_j),\ E(i,j),\ F(i,j)\}. \qquad (2)$$

In view of the mosaic nature of molecular sequences, the sequence alignment problem of current interest is usually that of finding significant alignments of segments (contiguous subsequences) of the two sequences. Such alignments are known as local alignments. Smith and Waterman (1981) showed that a simple modification of (1) gives a solution to the local alignment problem. The local alignment problem is to find all alignments which have score:

$$H(\mathbf{a},\mathbf{b})=\max\{S(a_ua_{u+1} \ldots a_v, b_xb_{x+1} \ldots b_y):\ 1\leqslant u\leqslant v\leqslant n,\ 1\leqslant x\leqslant y\leqslant m\}.$$

The algorithm can be obtained from $H(0,0)=H(i,0)=H(0,j)=0$ for $1\leqslant i\leqslant n$, $1\leqslant j\leqslant m$, and:

$$H(i,j)=\max\{H(i-1,j-1)+s(a_i,b_j),\ \max_{1\leqslant k\leqslant i}\{H(i-k,j)+w(k)\},$$
$$\max_{1\leqslant l\leqslant j}\{H(i,j-l)+w(l)\},\ 0\}. \qquad (3)$$

The $O(n^2)$ improvement for $w(k)=\alpha+\beta k$ yields:

$$E(i,j)=\max\{E(i-1,j)+\beta,\ H(i-1,j)+\alpha+\beta\}$$
$$F(i,j)=\max\{F(i,j-1)+\beta,\ H(i,j-1)+\alpha+\beta\}$$
$$H(i,j)=\max\{H(i-j,j-1)+s(a_i,b_j),\ E(i,j),\ F(i,j),\ 0\}. \qquad (4)$$

Of course $H(\mathbf{a},\mathbf{b})=\max\{H(i,j):\ 1\leqslant i\leqslant n,\ 1\leqslant j\leqslant m\}$.

In Waterman and Eggert (1987) a method was introduced to produce the 2nd, 3rd, . . . , Kth best segment or local alignments as well as the best. The succeeding alignments are not allowed to have aligned pairs (a_i/b_j) in common. The computing time of the Waterman and Eggert method is $O(n^2+\sum_{l=1}^{K} L_l^2)$, where L_l is the length of the lth best alignment. Since we can compute as many non-intersecting alignments as desired, it is necessary to have an explicit criterion for accepting an alignment. Therefore, we now turn to a discussion of the statistical distribution of alignment scores.

It is a cliché that our ability to compute exceeds our ability to understand. In the present context, we can compute alignments that look good to us but might be no better than expected from random sequences. Since the goal of sequence alignment is to find interesting biology, it is desirable to screen out alignments that are not statistically significant. The solution is often Monte Carlo simulation. Some progress has been made on theoretical aspects of the problem (Waterman *et al.*, 1987; Arratia *et al.*, 1990). Detailed results are known for the length of the longest run of identical letters (matches) as well as the longest match with r mismatches. Recently this work has been generalized to the distribution of the longest run with α proportion of matches where $\alpha > P(\text{match})$ (Arratia *et al.*, 1990). This work rests on recent advances in Poisson approximation (Arratia *et al.*, 1989). There are two lines of study for local algorithms with general scoring schemes. Arratia *et al.* (1988) provided strong laws for the score as well as the statistical distribution of letters in the alignment. Karlin and Altschul (1990) give the results for more general scoring schemes as well as a Poisson approximation to assess statistical significance. Another approach appears in Waterman *et al.* (1987). For:

$$s(a,b)=\begin{cases}+1, & a=b\\ -\mu, & a\neq b\end{cases}$$

and $w(k) = -\delta k$, it is known that, except for points on a curve in (μ, δ), either:

$$P\left(\lim_{n\to\infty} H(n,n)/n = c\right) = 1$$

or

$$P\left(\lim_{n\to\infty} H(n,n)/\log(n) = d\right) = 1.$$

The idea is that whenever μ and δ are "large enough" $H(n, n)$ behaves like

$\log(n)$. The results on score distribution provide us with a rich if still incomplete theory for determining statistical significance of alignment scores.

The Poisson approximation arises in the following way. Set a test value t and compute alignments satisfying $\{H>t\}$. The Waterman and Eggert algorithm for the K best alignments requires the alignments to be disjoint; in the language of Arratia *et al.*, the events are declumped. Let Z equal the number of alignments $\{H>t\}$ produced by this method, and let $\lambda=E(Z)$. Order the scores by size:

$$H_{(Z)}\leqslant H_{(Z-1)}\leqslant \cdots \leqslant H_{(1)}.$$

Define:

$$T=\max\{S(x_1x_2\ldots x_i, y_1, y_2\ldots y_j):\quad i\leqslant n,\quad j\leqslant m\}.$$

The first Poisson approximation result states:

$$P(H_{(1)}\leqslant x)\approx e^{-\lambda P(T>x)}.$$

The remaining order statistics have approximate distribution:

$$P(H_{(j)}\leqslant x)\approx e^{-\lambda P(T>x)}\sum_{i=0}^{j-1}\frac{(\lambda P(T>x))^i}{i!}.$$

This result can be rigorously proved for the case of no indels and no mismatches. See Goldstein and Waterman (1992) for a generalization to mismatches. We have presented these results here since we use the largest scores in our new algorithm. Below we compute the K best local alignments as input to our new algorithm. Since we cannot estimate $P(T>x)$, the formula for $P(H_{(j)}\leqslant x)$ cannot be used directly.

3. Inversions. Our first goal is to describe a dynamic programming algorithm for optimal alignment of two DNA sequences with substitutions, insertions, deletions and inversions. An inversion of a DNA sequence is defined to be the reverse complement of the sequence. While the number of inversions is not restricted, the inversions will not be allowed to intersect one another. Later we will discuss the case of intersecting inversions. While we could describe other versions of our algorithm, including full or global sequence alignment, here we present the local alignment algorithm with a linear gap weighting function. This is probably the most useful version of the method.

When we allow inversions, we must realize that the inverted regions will not exactly match and must themselves be aligned. In addition one of the inverted regions must be complemented to preserve the polarity of the DNA sequence. Let us define:

$$Z(g, h; i, j) = S_1(a_g a_{g+1} \ldots a_i, \bar{b}_j \bar{b}_{j-1} \ldots \bar{b}_h),$$

where $\bar{A} = T$, $\bar{C} = G$, $\bar{G} = C$, and $\bar{T} = A$. In the original sequences the segments $a_g a_{g+1} \ldots a_i$ and $b_h b_{h+1} \ldots b_j$ are matched after an inversion. This means that $a_g a_{g+1} \ldots a_i$ and $\bar{b}_j \bar{b}_{j-1} \ldots \bar{b}_h$ are aligned. $Z(g, h; i, j)$ is indexed by the beginning (g, h) and ending (i, j) coordinates of the sequences in their original order. The function S_1 is the alignment score defined in equation (2) using the matching function $s_1(a, b)$, and gap function $w_1(k) = \alpha_1 + \beta_1 k$. Each inversion is charged an additional cost γ. The non-inverted alignment uses matching function $s_2(a, b)$ and gap function $w_2(k) = \alpha_2 + \beta_2 k$.

The recursion for the best score W, with inversions, is given in the following.

```
Algorithm: All inversions.
set $U(i,j) = V(i,j) = W(i,j) = 0$ if $i = 0$ or $j = 0$.
for $j = 1$ to $m$,
    for $i = 1$ to $n$.
        {   $U(i,j) = \max\{U(i-1,j) + \beta_2,\ W(i-1,j) + \alpha_2 + \beta_2\}$
            $V(i,j) = \max\{V(i,j-1) + \beta_\beta,\ W(i,j-1) + \alpha_2 + \beta_2\}$
            for $h = j$ to 1
                for $g = i$ to 1
                    compute $Z(g, h; i, j)$.
            $W(i,j) = \max\{\max_{1 \leq g \leq i,\ 1 \leq h \leq j} \{W(g-1, h-1) + Z(g, h; i, j)\} + \gamma,$
$W(i,-1, j-1) + s_2(a_i, b_j),\ U(i,j),\ V(i,j),\ 0\}$.      (5)
        }
best inversion score $= \max\{W(i,j)\colon 1 \leq i \leq n,\ 1 \leq j \leq m\}$.
```

The proof that the recursion gives the optimal score for non-intersecting inversions follows the usual proof for sequence alignment (Theorem 1, Waterman, 1989).

The system of recurrences in (5) is expensive in computation time. If $Z(g, h; i, j)$ is computed for each (g, h) where $1 \leq g \leq i$ and $1 \leq h \leq j$, this takes time $O(i^2 j^2)$, and the full algorithm (5) takes time $O(n^6)$, when $n = m$. If general $w(k)$ is used, the corresponding version of our algorithm (5) takes time $O(n^7)$, when $n = m$. By storing values of $Z(g, h; i, j)$ the "compute $Z(g, h; i, j)$" step can be done in time $O(1)$, so that the full algorithm (5) can be executed in time $O(n^4)$ and space $O(n^2)$.

Clearly the algorithm (5) is too costly in time for any problem of interest. Essentially the reason for this is that many very poor quality inversions are calculated and rejected. Biologists will only be interested in longer, high quality inversions (Howe *et al.*, 1988; Zhou *et al.*, 1988). Fortunately there is a computationally efficient way to choose these inversions and dramatically speed up the alignment algorithm.

We first apply the local algorithm with $s_1(a, b)$ and $w_1(k)=\alpha_1+\beta_1 k$ to the sequences $\mathbf{a}=a_1 \ldots a_n$ and the inverted sequence $\mathbf{b}^{(\mathrm{inv})}=\bar{b}_m\bar{b}_{m-1} \ldots \bar{b}_1$. The extension of this algorithm from Waterman and Eggert (1987) gives the K best (inversion) local alignments with the property that no match or mismatch is used more than once in the alignments. Each time a best alignment is located the matrix must be recalculated to remove the effect of the alignment. If alignment i has length L_i, the time required to produce the list $\mathscr{L}$ of the K best inversion alignments is $O(nm+\sum_{i=1}^{K} L_i^2)$. To reduce the time requirement we chose an appropriate value of K. To further reduce running time we could impose a score threshold C_1 chosen so that the probability two random sequences have a Kth best alignment score $\geqslant C_1$ is small. See the discussion of statistical significance in Section 2.

Algorithm; Best inversions.

```
(I)
apply Waterman–Eggert algorithm to $\mathbf{a}$ and $\mathbf{b}^{(\mathrm{inv})}$.
        $\mathscr{L}=\{(Z(g, h; i, j), (g, h), (i, j)): K \text{ best}\}$
(II)
set $U(i,j)=V(i,j)=W(i,j)=0$ if $i=0$ or $j=0$.
for $j=1$ to $m$
    for $i=1$ to $n$
        { $U(i,j)=\max(U(i-1,j)+\beta_2,\ W(i-1,j)+\alpha_2+\beta_2\}$
          $V(i,j)=\max\{V(i,j-1)+\beta_2,\ W(i,j-1)+\alpha_2+\beta_2\}$
          $W(i,j)=\max\{\max_{\mathscr{L}}\{W(g-1,h-1)+Z(g,h;i,j)\}+\gamma,$      (6)
        }        $W(i-1,j-1)+s_2(a_i,b_j),\ U(i,j),\ V(i,j),\ 0\}$.
best inversion score $=\max\{W(i,j): 1\leqslant i\leqslant n,\ 1\leqslant j\leqslant m\}$.
```

We have greatly reduced computation time. (I) of the algorithm can be done in time $O(nm)$ by (2). To execute (II) requires time proportional to nm times a constant plus the average number of elements in $\mathscr{L}$ that "end" at (i, j). It might seem that only one best inversion has this property, but recall that we align $a_g a_{g+1} \ldots a_i$ with $\bar{b}_j\bar{b}_{j-1} \ldots \bar{b}_h$. This allows the possibility of several elements with "end" (i, j). Still the list $\mathscr{L}$ is restricted to K elements. In the illustrative example discussed next $|\mathscr{L}|=2$. Clearly part II of the algorithm best inversions runs in time $O(nm+\sum_{i=1}^{|\mathscr{L}|} L_i^2)$.

For maximum flexibility we have allowed $s_1(a, b)$ and $s_2(a, b)$ as well as $w_1(k)$ and $w_2(k)$ to have different values. This might be advisable if the inversion segments are thought to have evolved differently from the rest of the alignment. We have experimented with these parameters with the example of Section 5 and did not improve results over $s_1=s_2$ and $w_1=w_2$. In general, values of $s(a, b)$ and $w(k)=\alpha k+\beta$ are chosen in an *ad hoc* fashion and by experience for the standard alignment algorithms. More than twice as many parameters does not make that task easier. We recommend setting $s_1 = s_2$ and $w_1 = w_2$. Still, γ and $|\mathscr{L}|$ are new parameters for the inversion algorithm. We recommend choice of $\gamma \geqslant w(1)$ and $|\mathscr{L}|$ as large as is computationally feasible.

To illustrate our algorithm we use the sequences $\mathbf{a}$ = CCAATCTAC-TACTGCTTGCA and $\mathbf{b}$ = GCCACTCTCGCTGTACTGTG. The matching functions are:

$$s_1(a, b) = s_2(a, b) = \begin{cases} 10 & \text{when } a = b \\ -11 & \text{when } a \neq b \end{cases}$$

while:

$$w_1(k) = w_2(k) = -15 - 5k.$$

The inversion penalty is $\gamma = -2$ while the list $\mathscr{L}$ is defined by $K = 2$. The two alignments in $\mathscr{L}$ are shown in Fig. 1a where the matched-mismatched pairs are boxed. The best local alignment with inversion is shown in Fig. 1b. Figure 2 is a schematic of Fig. 1. In Fig. 2a, 1 and 2 correspond to the best and second best alignments, respectively. In Fig. 2b, 0 denotes alignment without inversion, while 1 denotes alignment of inverted regions.

4. The *J* Best Alignments with Inversions. The algorithm for local alignment with inversions allows us to find all alignments with the optimal score. Recall that the Waterman and Eggert algorithm produces the K best local alignments with the property that none of these alignments share a pair of matched (or mismatched) letters. We have of course utilized that algorithm to produce list $\mathscr{L}$, $|\mathscr{L}| = K$, which reduces computing time for alignment with inversions. In this section we will describe a method similar to the Waterman and Eggert algorithm for producing the J best alignments with inversions.

Next we review in some detail the Waterman and Eggert (1987) algorithm. In that paper technical details are given so that one optimal alignment is chosen at each stage $(1, 2, \ldots)$. Occasionally many alternate alignments have the same score; they often differ by small details. One guiding principle is to choose an alignment of shortest length. This is relevant to our algorithm for inversions since different alignment choices in $\mathscr{L}$ can in some cases create different overall alignments.

Having chosen an alignment, the next task is to recompute the matrix $H(i, j)$ so that a new matrix $H^*(i, j)$ is defined to be the matrix obtained not allowing any match (mismatch) from the alignments already output. The entire matrix H can be recomputed to obtain H^* in time $O(nm)$ but this is not necessary. The upper left most alignment pair (p, q) is the first value that changes:

$$H^*(p, q) = \max\{E(p, q), F(p, q), 0\}.$$

	C	A	C	A	G	T	A	C	A	G	C	G	A	G	A	G	T	G	G	C
C	10	0	10	0	0	0	0	10	0	0	10	0	0	0	0	0	0	0	0	10
C	10	0	10	0	0	0	0	10	0	0	10	0	0	0	0	0	0	0	0	10
A	0	20	0	20	0	0	10	0	20	0	0	0	10	0	10	0	0	0	0	0
A	0	10	9	10	9	0	10	0	10	9	0	0	10	0	10	0	0	0	0	0
T	0	0	0	0	0	19	0	0	0	0	0	0	0	0	0	0	10	0	0	0
C	10	0	10	0	0	0	8	10	0	0	10	0	0	0	0	0	0	0	0	10
T	0	0	0	0	0	$\boxed{10}$	0	0	0	0	0	0	0	0	0	0	10	0	0	0
A	0	10	0	10	0	0	$\boxed{20}$	0	10	0	0	0	10	0	10	0	0	0	0	0
C	10	0	20	0	0	0	0	$\boxed{30}$	10	5	10	0	0	0	0	0	0	0	0	10
T	0	0	0	9	0	$\boxed{10}$	0	10	19	0	0	0	0	0	0	0	10	0	0	0
A	0	10	0	10	0	0	$\boxed{20}$	5	20	8	0	0	10	0	10	0	0	0	0	0
C	10	0	20	0	0	0	0	$\boxed{30}$	10	9	18	0	0	0	0	0	0	0	0	10
T	0	0	0	9	0	10	0	10	$\boxed{19}$	0	0	7	0	0	0	0	10	0	0	0
G	0	0	0	0	19	0	0	5	0	$\boxed{29}$	9	10	0	10	0	10	0	20	10	0
C	10	0	10	0	0	8	0	10	0	9	$\boxed{39}$	19	14	9	4	0	0	0	9	20
T	0	0	0	0	0	10	0	0	0	4	19	28	8	3	0	0	10	0	0	0
T	0	0	0	0	0	10	0	0	0	0	14	8	17	0	0	0	10	0	0	0
G	0	0	0	0	10	0	0	0	0	10	9	24	4	27	7	10	0	20	10	0
C	10	0	10	0	0	0	0	10	0	0	20	4	13	7	16	0	0	0	9	20
A	0	20	0	20	0	0	10	0	20	0	0	9	14	2	17	5	0	0	0	0

```
Inversion #1: g:10 h:10 i:15 j:15 Z:39
TACTGC
||| ||
TACAGC

Inversion #2: g: 7 h:13 i: 9 j:15 Z:30
TAC
|||
TAC
```

	G	C	C	A	C	T	C	T	C	G	C	T	G	T	A	C	T	G	T	G
C	0	$\boxed{10}$	10	0	10	0	10	0	10	0	10	0	0	0	0	10	0	0	0	0
C	0	10	$\boxed{20}$	0	10	0	10	0	10	0	10	0	0	0	0	10	0	0	0	0
A	0	0	0	$\boxed{30}$	10	5	0	0	0	0	0	0	0	0	10	0	0	0	0	0
A	0	0	0	10	$\boxed{19}$	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0
T	0	0	0	5	0	$\boxed{29}$	9	10	0	0	0	10	0	10	0	0	10	0	10	0
C	0	10	10	0	15	9	$\boxed{39}$	19	20	9	10	0	0	0	0	10	0	0	0	0
T	0	0	0	0	0	25	19	$\boxed{49}$	29	24	19	20	9	10	0	0	20	0	10	0
A	0	0	0	10	0	5	14	$\boxed{29}$	38	18	13	8	9	0	20	0	0	9	0	0
C	0	10	10	0	20	0	15	24	$\boxed{39}$	27	28	9	4	0	28	30	10	5	0	0
T	0	0	0	0	0	30	10	25	19	28	16	38	18	14	8	17	40	20	15	10
A	0	0	0	10	0	10	19	14	14	8	17	18	27	7	24	5	20	29	9	4
C	0	10	10	0	20	5	20	9	24	4	18	13	7	16	4	34	15	9	18	0
T	0	0	0	0	0	30	10	30	10	13	0	28	8	17	5	14	44	24	19	14
G	10	0	0	0	0	10	19	10	19	20	2	8	38	18	13	9	24	54	34	29
C	0	20	10	0	10	5	20	8	20	8	30	10	18	27	76	56	51	46	43	36
T	0	0	9	0	0	20	0	30	10	9	10	40	20	28	56	$\boxed{65}$	66	46	56	36
T	0	0	0	0	0	10	9	10	19	0	5	20	29	30	51	45	$\boxed{75}$	55	56	45
G	10	0	0	0	0	0	0	5	0	29	9	15	30	18	46	40	55	$\boxed{85}$	65	66
C	0	20	10	0	10	0	10	0	15	9	39	19	14	19	41	56	50	65	74	54
A	0	0	9	20	0	0	0	0	0	4	19	28	8	3	36	36	45	60	54	63

```
used: Inversion #1
i:18 j:18 W:85
CCAATCTAC******TTG
||| ||| |iiiiii ||
CCACTCT-C******CTG
```

Figure 1. Best local alignment with inversions of **a** = CCAATCTACTACTGCTTGCA and **b** = GCCACTCTCGCTGTACTGTG. (a) Shows the matrix H for **a** and $\mathbf{b}^{(\mathrm{inv})}$ and the two alignments in $\mathscr{L}$. (b) Shows matrix W and the best alignment.

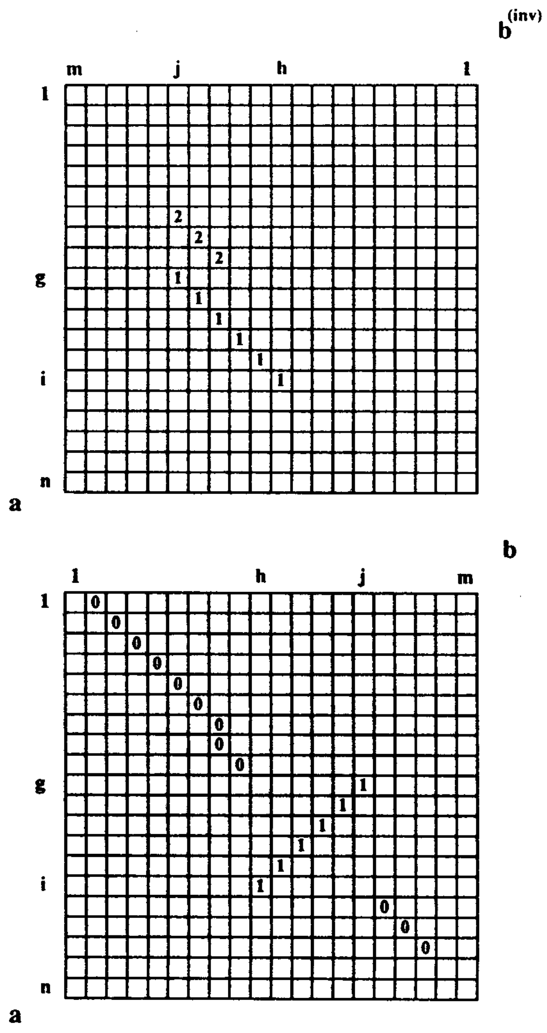

Figure 2. A schematic of the output presented in Fig. 1.

$H(p-1, q-1)+s(a_p, b_q)$ is eliminated from the maximum since a_p/b_q is in the alignment. The computation proceeds along the pth row from column q to column r until:

$$H^*(p, r)=H(p, r),$$

$$E^*(p, r)=E(p, r),$$

and

$$F^*(p, r)=F(p, r). \tag{7}$$

From this point on these equations must continue to hold. Then we perform the same operations on the qth column. We proceed by induction, switching from rows to columns until the effect of the alignment on H^* has been calculated. The time to perform these operations on an alignment of length L is approximately proportional to L^2.

We now turn to modifying our algorithm for inversions to yield the J best alignments. Our object is to produce the J best alignments that do not share a match, mismatch or inversion. Therefore when an inversion from $\mathscr{L}$ is used in an alignment, it cannot be used in a succeeding alignment. This is accomplished by appropriately changing $\mathscr{L}$ as the algorithm proceeds. As in (7) the computation proceeds along the pth row from column q to column r until

$$U^*(p,r)=U(p,r)$$
$$V^*(p,r)=V(p,r)$$

and

$$W^*(p,r)=W(p,r). \tag{8}$$

Then the computations are performed on the qth column. In this way an isolated island of the matrix is recalculated.

A second, more complicated feature of the algorithm arises at this point. The validity of the recalculation procedure in (8) is justified by the basic property of the dynamic programming recursion equations (6). However these recursions allow $W(g-1,h-1)$ to affect $W(i,j)$ if $(Z(g,h;i,j),(g,h),(i,j))\in\mathscr{L}$. Therefore after an island of recalculation is performed, we examine $\mathscr{L}$ for (i,j) such that the value $W^*(i,j)$ might be changed. That is, where $W^*(g,h)\neq W(g,h)$. Finding such an entry begins a new island of recalculation. Therefore the procedure of recalculation can initiate a cascade of such islands. In Fig. 3 we give a schematic illustration with two islands. Matrix entries of * indicate that at least one equation in (8) does not hold. The inversion joining the islands is denoted by 1's. If $\mathscr{L}$ has a large number of entries, the likelihood of new islands is increased. It is therefore difficult to give a rigorous analysis of the running time of finding the J best alignments. If $\mathscr{L}$ is short, it remains $O(nm+\sum_{i=1}^{J} L_i^2)$. Including the running time for computing $\mathscr{L}$ adds $O(nm+\sum_{i=1}^{K} L_{J+i}^2)$.

5. Example. We now apply the new algorithm to a biological example. In the work of Clary and Wolstenholme (1985) the mitochondrial genomes of vertebrates are compared to the mitochondrial genome of *Drosophila yakuba*. The mitochondria of vertebrates contain an inverted URF6 gene relative to the *Drosophila* mitochondrial genome. For our comparison we use mitochondrial DNA from *D. yakuba* (GenBank entry DRYMTCG, using nucleotides 9987–11651) and mouse (GenBank entry MUSMT, using nucleotides

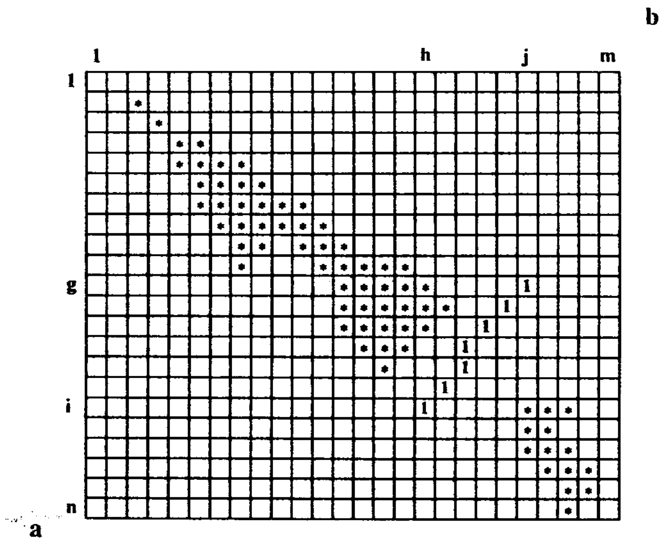

Figure 3. Schematic illustration of islands of recalculation.

13 546–15 282). The putative organization of genes in the sequences is as follows:

	Drosophila yakuba	mouse	
URF6	1–525	519–1	(inverted)
tRNA Glu		588–520	(inverted)
cytochrome b	529–1665	594–1737.	

(Here position 1 in *D. yakuba* corresponds to GenBank position 9 987 and position 1 in mouse corresponds to GenBank position 13 546.) This is a difficult alignment problem, due to rapid evolution of mitochondrial genomes.

The scoring functions are:

$$s_1(a, b) = s_2(a, b) = \begin{cases} +10 & a = b \\ -9 & a \neq b \end{cases}$$

$$w_1(k) = w_2(k) = -15 - 5k,$$

and

$$\gamma = -20.$$

For $|\mathscr{L}| = 400$, step I of the algorithm forms the list $\mathscr{L}$ of potential inversions. Inversion No. 26 (in order of score) is the one ultimately used in the alignment. We show 100 of the inversion alignment locations by straight lines in Fig. 4a. While alignment No. 26 is shorter than most of others, it has approximately the same 40% identity as the other, longer inversion alignments. The best local alignment with inversions is shown schematically in Fig. 4b. Inversion No. 26 appears as an antidiagonal straight line at the upper left corner of the square—the inversion is at the 5′ end of our sequences.

The alignment relates fairly well to the gene organization reported above. Inversion No. 26 used in the alignment aligns positions 7–480 in *D. yakuba* to positions 58–542 in mouse. This is about 45 less bases of *D. yakuba* and an additional 20 bases of mouse than indicated in the gene organization table.

A change of the scoring functions by increasing the deletion penalties to $w_1(k) = w_2(k) = 20k$ and holding the other values fixed, changes the answer by eliminating the inversion from the best local alignment. This comes entirely from the increased cost of indels.

We feel this example illustrates the power of the new algorithm. Even with a long list of candidate inversions with larger scores, our program was able to detect what is apparently the biologically correct inversion. With a larger penalty for indels, the program fails on this difficult alignment problem.

(a)

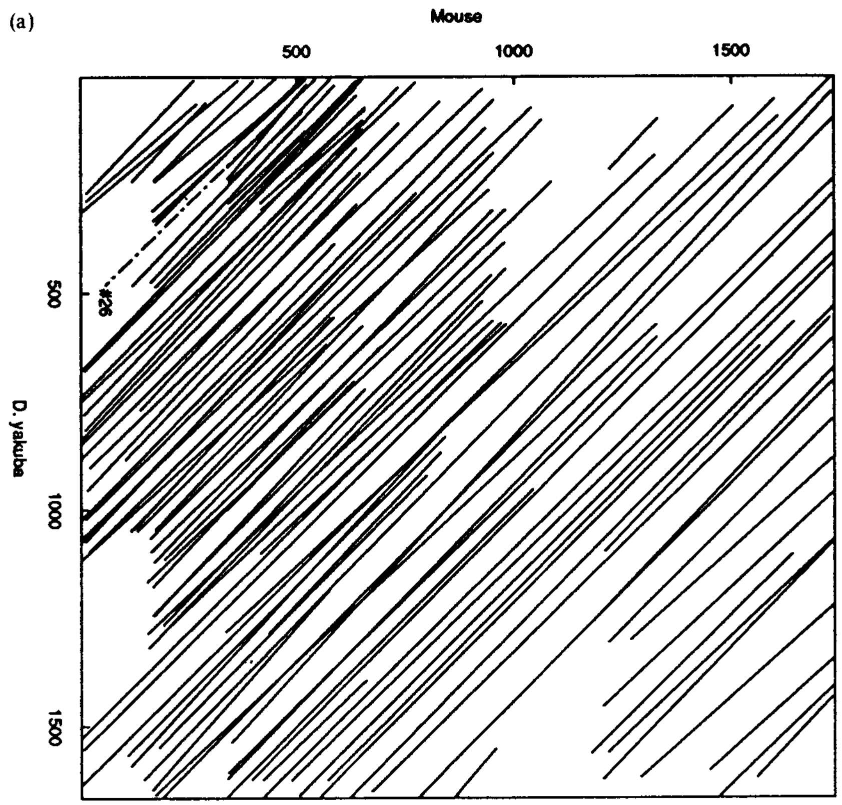

Figure 4(a).

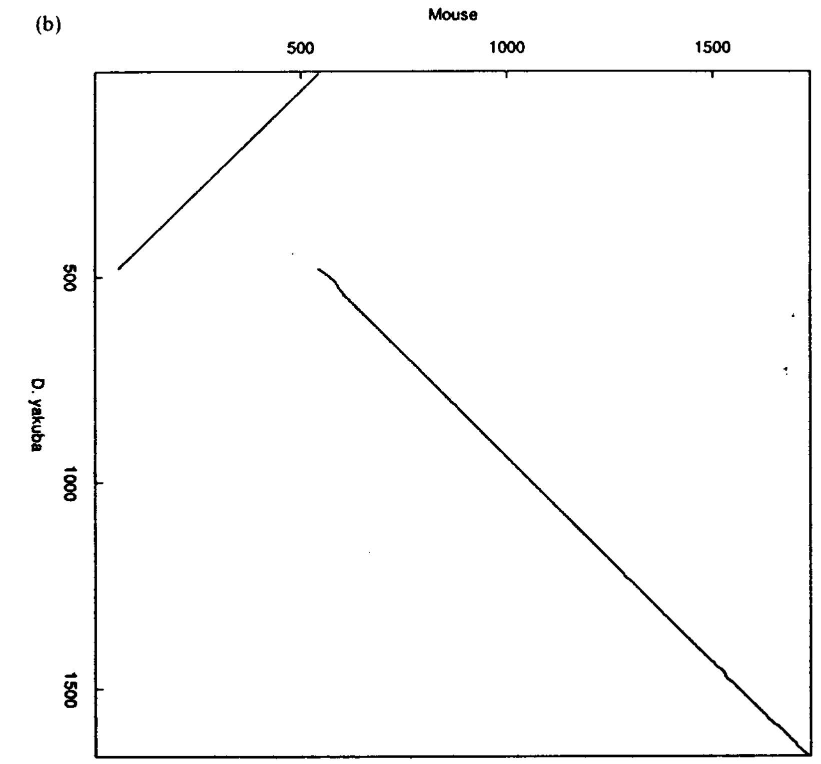

Figure 4. Inversion alignment algorithm applied to *Drosophilia*–mouse sequences. (a) is a schematic of the candidate inversions. (b) is a schematic of the optimal local alignment with inversions.

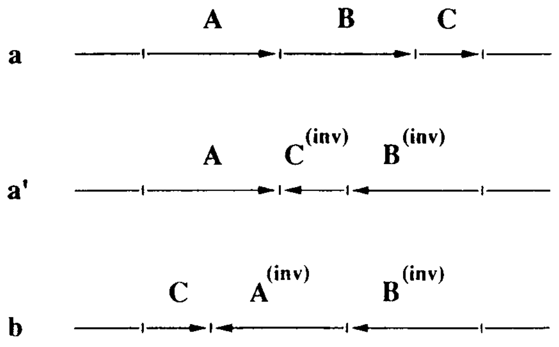

Figure 5. An illustration of overlapping inversions.

6. Extensions. Our algorithm for inversions avoids the computational difficulties of past algorithms by omitting short inversions. In addition, to obtain our solution we restricted ourselves to any number of non-intersecting inversions. In the course of the evolution, it is likely that inversions can overlap. Figure 5 provides a schematic of the effect of such events. To proceed from **a** to **a**′, the segment **BC** is inverted to become $\mathbf{C}^{(\mathrm{inv})}\mathbf{B}^{(\mathrm{inv})}$. Then the **a**′ segment $\mathbf{AC}^{(\mathrm{inv})}$ is itself inverted to become $\mathbf{CA}^{(\mathrm{inv})}$ in **b**. The effect of this ($\mathbf{a} \rightarrow \mathbf{b}$) is inversion of the segments **A** and **B**, and the translocation of **C** from 3′ of **B** in **a** to 5′ of $\mathbf{A}^{(\mathrm{inv})}$ in **b**. Given our method of constructing $\mathscr{L}$, the inversion list, it is likely that even if $\mathbf{A}^{(\mathrm{inv})}$ and $\mathbf{B}^{(\mathrm{inv})}$ are statistically significant and can be found in $\mathscr{L}$, they will not match exactly, at their ends. If the alignment of **A** with $\mathbf{A}^{(\mathrm{inv})}$ were to extend into **B**, we might not find these adjacent inversions. While we could develop an algorithm to search for inversions with overlaps such as we describe in Fig. 5, we have not done so. Higher order intersecting events significantly complicate the situation, and it remains an open problem to devise practical algorithms to handle these cases.

This research was supported by the Deutsche Forschungsgemeinschaft, the National Science Foundation, and the National Institutes of Health. We are grateful to Mark Eggert for computing assistance and to Gary Churchill and Arndt von Haeseler for helpful discussions.

LITERATURE

Arratia, R., P. Morris and M. S. Waterman. 1988. Stochastic scrabble: a law of large numbers for sequence matching with scores. *J. appl. Prob.* **25**, 106–119.

Arratia, R. A., L. Goldstein and L. Gordon. 1989. Two moments suffice for Poisson approximation: The Chen–Stein method. *Annls Prob.* **17**, 9–25.

Arratia, R. A., L. Gordon and M. S. Waterman. 1990. The Erdös-Rényi law in distribution, for coin tossing and sequence matching. *Annls Statist.* **18**, 539–570.

Clary, D. O. and D. R. Wolstenholme. 1985. The mitochondrial DNA molecule of *Drosophila yakuba*: nucleotide sequence, gene organization, and genetic code. *J. molec. Evol.* **22**, 252–271.

Goldstein, L. and M. S. Waterman. 1992. Poisson, compound Poisson, and process approximations for testing statistical significance in sequence comparisons. *Bull. math. Biol.* in press.

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

Howe, C. J., R. F. Barker, C. M. Bowman and T. A. Dyer. 1988. Common features of three inversions in wheat chloroplast DNA. *Curr. Genet.* **13**, 343–349.

Karlin, S. and S. F. Altschul. 1990. Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes. *Proc. natn. Acad. Sci. U.S.A.* **87**, 2264–2268.

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

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

Smith, T. F. and M. S. Waterman. 1981. Identification of common molecular subsequences. *J. molec. Biol.* **147**, 195–197.

Smith, T. F., M. S. Waterman and W. M. Fitch. 1981. Comparative biosequence metrics, *J. molec. Evol.* **18**, 38–46.

Wagner, R. A. 1983. On the complexity of the extended string-to-string correction problem. In *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison.* D. Sankoff and J. B. Kruskal (Eds), pp. 215–235. London: Addison-Wesley.

Waterman, M. S., T. F. Smith and W. A. Beyer. 1976. Some biological sequence metrics. *Adv. Math.* **20**, 367–387.

Waterman, M. S. 1984. General methods of sequence comparison. *Bull. math. Biol.* **46**, 473–500.

Waterman, M. S. and M. Eggert. 1987. A new algorithm for best subsequence alignments with application to tRNA–rRNA comparisons. *J. molec. Biol.* **197**, 723–728.

Waterman, M. S., L. Gordon and R. Arratia. 1987. Phase transitions in sequence matches and nucleic acid structure. *Proc. natn. Acad. Sci. U.S.A.* **84**, 1239–1243.

Waterman, M. S. 1989. Sequence alignments. In *Mathematical Methods for DNA Sequences*, M. S. Waterman (Ed.), pp. 53–92. Boca Raton, Florida: CRC Press.

Zhou, D. X., O. Massenet, F. Quigley, M. J. Marion, F. Monéger, P. Huber and R. Mache. 1988. Characterization of a large inversion in the spinach chloroplast genome relative to *Marchantia*: a possible transposon-mediated origin. *Curr. Genet.* **13**, 433–439.

Received 5 August 1990
Revised 18 March 1991