Pergamon

Bulletin of Mathematical Biology Vol. 56, No. 4, pp. 743–767, 1994
Elsevier Science Ltd

0092-8240/94 $7.00 + 0.00

0092-8240(93)E0001-E

PARAMETRIC AND ENSEMBLE SEQUENCE ALIGNMENT ALGORITHMS

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

Recently algorithms for parametric alignment (Waterman *et al.*, 1992, *Natl Acad. Sci. USA* **89**, 6090–6093; Gusfield *et al.*, 1992, *Proceedings of the Third Annual ACM-SIAM Discrete Algorithms*) find optimal scores for all penalty parameters, both for global and local sequence alignment. This paper reviews those techniques. Then in the main part of this paper dynamic programming methods are used to compute ensemble alignment, finding all alignment scores for all parameters. Both global and local ensemble alignments are studied, and parametric alignment is used to compute near optimal ensemble alignments.

1. Introduction. DNA sequencing has precipitated a revolution in biology. Rapid sequencing techniques were introduced a little more than a decade ago and the rate of sequencing continues to accelerate. As of Winter 1993 the international databanks (EMBL, GenBank and DDBJ) contain approximately 50×10^6 base pairs of DNA sequences with an average sequence length of about 1000 base pairs. Improvements in sequencing technology continue to be made and the associated discoveries in biology are staggering. Recently there has been an exciting new initiative in molecular biology, the human genome project. This project has as its objective the characterization of an entire human genome of 3×10^9 base pairs as well as genomes of other model organisms. Even though the basic goal of the genome project is very important and will take years to accomplish, the project has much broader implications and could be the basis of biology for the next century.

In this paper we will consider sequence alignment, one of the most utilized computational tools for the study of biomolecular sequences. Sequence alignments are used to detect relationships between sequences within and between species and have been the basis of fundamental insights into molecular biology. DNA and protein sequence alignments arrange two or more sequences, one written over another to indicate possible evolutionary relationships between the sequences. For purposes of exposition we will focus on DNA sequences of bases or nucleotides in this paper. Two aligned letters

might be identical or not. Aligned letters that are not identical indicate the substitution of one nucleotide for another. A letter aligned with a dash indicates the insertion or deletion (indel) of that letter. Most work on alignment algorithms is directed toward increasing the efficiency of existing methods or the derivation of new algorithms. Our study is aimed in a quite different direction, more in the spirit of statistical mechanics. We wish to study the set of all alignments.

Let us next look at a simple example where x = AAGTTC and y = AGCCC. An alignment $\mathbb{A}$ that might be suggested by the sequences is:

$$\begin{array}{c} A\ A\ G\ T\ T\ C \\ A - G\ C\ C\ C, \end{array}$$

where there are three identities, two substitutions or mismatches, and one indel. This alignment represents a specific hypothesis about the evolution of the sequences; three of the nucleotides have not changed since the common ancestor, there have been (at least) two substitutions, and one nucleotide has been either inserted or deleted. Sequences are aligned by computer in order to find good alignments. A computer is necessary because of the exponential number of possible alignments. More precisely, two sequences of length n and m can be aligned in $\binom{n+m}{n}$ ways. For example, two sequences of length 1000 have over 10^{600} distinct alignments (see Waterman, 1989).

There remains the important question of how to score an alignment. We present a simple heuristic derivation of a class of "alignment scores". If p is the probability of an identity, q the probability of a substitution, and r the probability of an indel, the above alignment has probability:

$$Pr = p^3 q^2 r^1.$$

Define score S' by the log likelihood:

$$S' = \log Pr = 3(\log p) + 2(\log q) + 1(\log r),$$

and recall that for global alignment:

$$2\#\text{identities} + 2\#\text{mismatches} + \#\text{indels} = n + m.$$

Define $S = S' - \frac{n+m}{2} \log s = S' - 11/2 \log s$, where s is a constant satisfying $\log(p/s) = 1$. We have simply subtracted from S' a constant depending on the sequence lengths and the base of the logarithm. S becomes:

$$S = 3 - 2\mu - 1\delta,$$

where $\mu = \log(s/q)$ and $\delta = \log(\sqrt{s}/r)$. Fortunately optimal score defined by:

$$G = \max\{\#\text{identities} - \mu\#\text{mismatches} - \delta\#\text{indels}\}$$

can be efficiently computed as we will see in the next section. It also has the simple maximum likelihood interpretation we have presented. We caution the reader that this simple likelihood reasoning is only a heuristic. See Tavaré (1986) for a related and mathematically rigorous discussion.

It is also of interest to convert the problem of alignment to a problem of molecular structures. Above $x=$AAGTTC and $y=$AGCCC are aligned by:

$$\begin{array}{c} \mathrm{A\,A\,G\,T\,T\,C} \\ \mathrm{A-G\,C\,C\,C.} \end{array}$$

To consider an analogous structure problem, complement the bases in $\mathbf{b}$ so that $\mathbf{b}^*=$TCGGG. What was an identity is now a base pair:

$$\begin{array}{c} \mathrm{A\,A\,G\,T\,T\,C} \\ \mathrm{T-C\,G\,G\,G.} \end{array}$$

We have 3 base pairs, 2 mispairs and 1 bulged base. Given alignment $\mathbb{A}$, the function:

$$S(\mathbb{A})=\#\text{basepairs}-\mu\#\text{mispairs}-\delta\#(\text{bulged bases})$$

is one of the simplest ways to assign energy (negative free energy) to a structure. Here μ and δ represent chemical potentials. In this elementary way, we can consider either alignments or structure, and to be definite the remainder of the paper will treat alignments.

It is natural to consider the ensemble $\mathscr{A}$ of alignments. Each alignment has score $S=a-b\mu-c\delta$, which is a real-valued function of (μ, δ) and the alignment:

$$S:\mathscr{A}\to\mathbf{R},$$

and where $a=$#identities, $b=$#mismatches and $c=$#indels.

At each fixed (μ, δ), we can view the set of all possible alignments $\mathscr{A}$ competing with one another. The dynamic programming algorithm efficiently computes the most energetically favorable or individually most likely alignments. In this paper we are going to study the set $\mathscr{S}=S(\mathscr{A})$, especially how this set of alignment scores and the optimal scores $G=\max\{S(\mathbb{A}):\mathbb{A}\in\mathscr{A}\}$ of S change as a function of (μ, δ). In statistical mechanics the temperature T is often a variable. In our study T is fixed and the chemical potentials (μ, δ) will vary. In statistical mechanics, it is of interest to compute the partition function:

$$Z=\sum_{\mathbb{A}} e^{S(\mathbb{A})/T}.$$

Below we will describe how to compute Z by finding all possible energies

$S = a - b\mu - c\delta$ and the frequencies $n(S)$ of each S. With knowledge of $S, n(S)$ we will have the ingredients for:

$$Z = \sum_S n(S)e^{S/T}.$$

While for global alignments it is possible to numerically compute Z at any fixed (μ, δ), we will symbolically compute Z for all (μ, δ) by finding all S and $n(S)$. For local alignments the corresponding numerical approach is much harder. The likelihood of S in the ensemble $\mathcal{S}(\mathcal{A})$ is:

$$\frac{n(S)e^{S/T}}{Z}.$$

Clearly a non-optimal value of S can have $n(S)$ large enough so that it is the most probable value of S.

While we have made this analogy or correspondence with statistical mechanics we will pursue it further in this paper. In an independent approach, Zhang and Marr (1993) have pursued the connections with statistical mechanics. See also Finkelstein and Roytberg (1993). Instead we want to look closely at the issues involved with computing $(n(S), S)$. The idea of using dynamic programming to compute partition functions has been around for a number of years. Temple Smith and I explored these ideas in connection with RNA secondary structure in the mid 1970s, eventually resulting in Howell *et al.* (1980). Instead of formal partition functions, we used generating functions that counted the structures with given numbers of base pairs, nearest neighbors, etc. . . . Recently a superb paper by McCaskill (1990) has appeared that carries those ideas much further. Two related papers by Thorne *et al.* (1991, 1992) study maximum likelihood alignment of DNA sequences. Parameter estimation is performed that accounts for all possible alignments. These ideas are of course related to the purely computational concerns of this paper.

Another set of related ideas arises in the area of near-optimal alignments. In Waterman (1983) and Byers and Waterman (1984) the traceback procedure for optimal alignments is modified to give all alignments within a fixed distance d of the optimal. Of course this set increases rapidly in size and is practical for only a small d. Later Vingron and Argos (1990) look at matching residues that belong to some near-optimal alignments, thereby giving a dot matrix view of the analysis. More recently, Naor and Brutlag (1993) has made a combinatorial analysis of near-optimal alignments.

In the next section on parametric alignment, we will review dynamic programming algorithms for global and local alignment and we will present a recent technique, parametric alignment, for finding the optimal scores for all

penalty parameters (μ, δ) along with some extensions. Then in Section 3 on ensemble alignment, we will present dynamic programming algorithms for computing all alignment hyperplanes and their frequencies for both global and local alignment. The number of alignment hyperplanes restricts the sequence length currently computationally feasible. The parametric optimal alignment algorithm is used in an algorithm that finds the alignment hyperplanes within d of the optimal, increasing the feasible sequence lengths somewhat. Our goal in this paper is to study the computation of $(n(S), S)$ for local and global alignment of DNA.

2. Parametric Sequence Alignment. As we have mentioned earlier, we restrict ourselves to the case where alignments are scored with 1 for an aligned pair of equal letters (identities $\equiv$ id), $-\mu$ for an aligned pair of unequal letters (mismatches $\equiv$ mm) and $-\delta$ for unaligned letters (insertions and deletions $\equiv$ indels). If we wished to apply these methods to protein sequences, we could shift the weight matrix, such as the Dayhoff matrix $D=(d_{ij})$, by μ: $D-\mu=(d_{ij}-\mu)$.

If $\mathscr{A}$ is the finite set of alignments for the sequences $\mathbf{x}=x_1 \ldots x_n$ and $\mathbf{y}=y_1 \ldots y_m$, define optimal score G by:

$$G=G(\mathbf{x}, \mathbf{y})\equiv\max\{\#\mathrm{id}-\mu\#\mathrm{mm}-\delta\#\mathrm{indels}\}.$$

Efficient dynamic programming algorithms to compute G for biological sequences in time $O(nm)$ can be traced to Needleman and Wunsch (1970). Surveys of more recent developments can be found in Waterman (1984, 1989). In this section we first review standard dynamic programming algorithms for optimal global (G) and local (H) alignment hyperplanes for fixed parameters (μ, δ). Then we give a parametric dynamic programming algorithm to obtain optimal alignment scores for all (μ, δ). Parametric dynamic programming has been studied in Fernandez-Baca and Srinivasan (1991), Gusfield *et al.* (1992) and Waterman *et al.* (1992).

2.1. *Dynamic programming alignment algorithms*

2.1.1. Optimal Global Alignment. The standard dynamic programming algorithm for computing global alignment scores $G(\mathbf{x}, \mathbf{y})=\max\{S(\mathbb{A}): \mathbb{A}\in\mathscr{A}\}$ goes as follows. Set:

$$G_{i,j}=G(x_1 \ldots x_i, y_1 \ldots y_j),$$

where $G_{i,j}=-(i+j)\delta$ if $i\cdot j=0$. The remaining $G_{i,j}$ can be computed recursively by:

$$G_{i,j}=\max\{G_{i-1,j-1}+s(x_i, y_j), G_{i-1,j}-\delta, G_{i,j-1}-\delta\}, \tag{1}$$

where

Table 1. Global alignment matrix, $(10 \times G_{i,j})$

		C	A	C	T	A	G
	0	−21	−42	−63	−84	−105	−126
A	−21	−9	−11	−32	−53	−74	−95
C	−42	−11	−18	−1	−22	−43	−64
C	−63	−32	−20	−8	−10	−31	−52
C	−84	−53	−41	−10	−17	−19	−40
T	−105	−74	−62	−31	0	−21	−28

$$s(x, y)=\begin{cases}1 & \text{if } x=y \\ -\mu & \text{if } x\neq y\end{cases}.$$

Since $G_{n,m}=G(\mathbf{x}, \mathbf{y})$, the alignment score can be computed in order $O(nm)$ steps. The function $s(x, y)$ can be made more general, but the algorithm corresponds to the definition of G above.

One way to obtain optimal alignments is to begin with the computed value of $G_{n,m}$ and traceback through the matrix. At each stage (i, j), the three cells $(i-1, j)$, $(i-1, j-1)$ and $(i, j-1)$ are checked to see whether the score $G_{i,j}$ can be obtained by using each of the corresponding matrix entries. If more than one possibility obtains, one is pursued and the remaining are placed on a stack for subsequent analysis. In this manner all optimal alignments can be obtained.

Table 1 shows the $(G_{i,j})$ matrix for aligning $\mathbf{x}=$ACCCT and $\mathbf{y}=$CACTAG for $(\mu, \delta)=(0.9, 2.1)$. The optimal alignment is:

C A C T A G
− A C C C T.

2.1.2. Optimal local alignment. The problem of local alignment is closely related to $G(\mathbf{x}, \mathbf{y})$, except that the score is optimized over all intervals or segments of x and y. The following algorithm first appeared in Smith and Waterman (1981). We define:

$$H(x, y)=\max\{G(x_i x_{i+1} \ldots x_k, y_j y_{j+1} \ldots y_l): 1\leqslant i\leqslant k\leqslant n,\ 1\leqslant j\leqslant l\leqslant m\}.$$

To find the score define:

$$H_{i,j}=H(x_1 \ldots x_i, y_1 \ldots y_j),$$

when $H_{i,j}=0$ whenever $i\cdot j=0$. The recursive calculation is now:

$$H_{i,j}=\max\{H_{i-1,j-1}+s(x_i, y_j), H_{i-1,j}-\delta, H_{i,j-1}-\delta, 0\}. \tag{2}$$

As above $H(\mathbf{x}, \mathbf{y})=\max_{i,j} H_{i,j}$ and the algorithm takes time $O(mn)$.

Table 2. Local alignment matrix, $(10 \times H_{i,j})$

	C	A	C	T	A	G
A	0	10	0	0	10	0
C	10	0	20	0	0	1
C	10	1	10	11	0	0
C	10	1	11	1	2	0
T	0	1	0	21	0	0

Alignments can be obtained from $(H_{i,j})$ in the same manner as from $(G_{i,j})$, except of course we traceback from $\max_{i,j} H_{i,j}$. Table 2 shows the $(H_{i,j})$ matrix for aligning $\mathbf{x} = \text{ACCCT}$ with $\mathbf{y} = \text{CACTAG}$ for $(\mu, \delta) = (0.9, 2.1)$. The optimal alignment is:

C A C T
C C C T.

2.2. *Parametric alignment.* Recently Waterman *et al.* (1992) and Gusfield *et al.* (1992) studied parametric alignment for global and local alignments. The goal of parametric alignment is to compute the score S for all values of the penalty parameters. The algorithms are valid for penalty functions that are linear in the parameters, such as the local and global alignment scores introduced in Section 2.1. We begin with a one-dimensional penalty function, and generalize to two- and three-dimensional problems. Our discussion for one and two-dimensions follows Waterman *et al.* (1992), with a new improvement (the "ε^*-method"). Gusfield (personal communication) has been able to perform parametric alignment in time proportional to $O(nm)$ times the number of cells in the tesselational for combinatorial objective functions such as we use here for illustration. In biology more complex objective functions often are required. There remains the question as to whether the "infinitesimals" we introduce give any advantage at all. In exploring that interesting question, the ε^*-method suggests there might be but proving its expected complexity requires a deeper statistical analysis than this author has been able to give.

2.2.1. One dimension. To define a one-dimensional problem, let $\mu = \mu_0 + \mu_1\lambda$ and $\delta = \delta_0 + \delta_1\lambda$ be linear functions of another parameter λ. Then:

$$\begin{aligned} S &= a - b\mu - c\delta \\ &= a - b\mu_0 - c\delta_0 - (b\mu_1 + c\delta_1)\lambda \\ &= a^* - b^*\lambda \end{aligned}$$

Table 3. Comparison matrix for $\lambda = 1-\varepsilon$

	c	t	g	t	c	g	c	t	g	c	a	c	g
t	0,0	1,0	0,0	1,0	0,0	0,0	0,0	1,0	0,0	0,0	0,0	0,0	0,0
g	0,0	0,0	2,0	0,2	0,1	1,0	0,0	0,0	2,0	0,2	0,0	0,0	1,0
c	1,0	0,0	0,2	1,1	1,2	0,0	2,0	0,2	0,2	3,0	1,2	1,0	0,0
c	1,0	0,1	0,0	0,0	2,1	0,3	1,0	1,1	0,0	1,2	2,1	2,2	0,4
g	0,0	0,1	1,1	0,0	0,3	3,1	1,3	0,1	2,1	0,3	0,3	1,2	3,2
t	0,0	1,0	0,0	2,1	0,3	1,3	2,2	2,3	0,5	1,2	0,0	0,0	1,4
g	0,0	0,0	2,0	0,3	1,2	1,3	0,4	1,3	3,3	1,5	0,3	0,0	1,0

and we can apply dynamic programming to find $H(\lambda)$ for any λ. In this way, every alignment $\mathbb{A}$ has a corresponding alignment line $a^* - b^*\lambda$ and $H(\lambda)$ is the maximum of all such lines at λ. The following lemma is elementary.

LEMMA 1. *Let* $\mathrm{H}(\lambda)$ *be the local alignment score for* x *and* y *with* $\mu = \mu_0 + \mu_1\lambda$ *and* $\delta = \delta_0 + \delta_1\lambda$. *Then* $\mathrm{H}(\lambda)$ *is a continuous, piecewise linear and decreasing function of* λ.

We now introduce infinitesimals to help us formulate the parametric algorithm. Let ε be a positive infinitesimal, that is $\varepsilon > 0$ is a small number such that any finite multiple remains smaller than any positive real. Our new numbers are written $u + v\varepsilon$, when $u, v \in R$. The usual lexicographic linear order is defined: Set $x = u_1 + v_1\varepsilon$ and $y = u_2 + v_2\varepsilon$. When $u_1 > u_2$, $x > y$. When $u_1 = u_2$ and $v_1 > v_2$, also $x > y$. Addition and scalar multiplication is defined in the usual way. The dynamic programming algorithms of Section 2.1 are defined when μ and δ are these new numbers, as only addition, subtraction and maximum are required. It is clear that if $S(u+v\varepsilon) = \alpha + \beta\varepsilon$, then $S(u) = \alpha$.

The reason for introducing infinitesimals is that several distinct alignment lines can be optimal for a specific parameter value λ. However $\lambda + \varepsilon$ necessarily lies in the interior of one of the piecewise linear segments of S and the optimal alignment line is therefore unique.

To illustrate this idea and the algorithm, two sequences are compared below in Table 3, where $H_{i,j} = u + v\varepsilon$ with u, v displayed in the matrix and $\lambda = 1-\varepsilon$. There are four local alignments, all with score 3, for $\lambda = 1$, $\mu = \lambda$ and $\delta = 2\lambda$. That is, for a identities, b mismatches and c indels, $S(\mathbb{A}) = a - (b+2c)\lambda$. Each of these alignments corresponds to one of the lines shown in Fig. 1. The optimal line at $\lambda = 1-\varepsilon$ has score $3+3\varepsilon$ and corresponds to the dominating line left of $\lambda = 1$.

The algorithm for one-dimensional problems is easily derived. We wish to compute $H(\lambda)$ for $\lambda \in [0, \infty]$. It is easy to find $H(0+\varepsilon)$, the optimal line intersecting $(0, H(0))$. Also $(\infty, H(\infty))$ is also easily determined by running the algorithm for H where we do not allow deletions or mismatches. Thus we begin

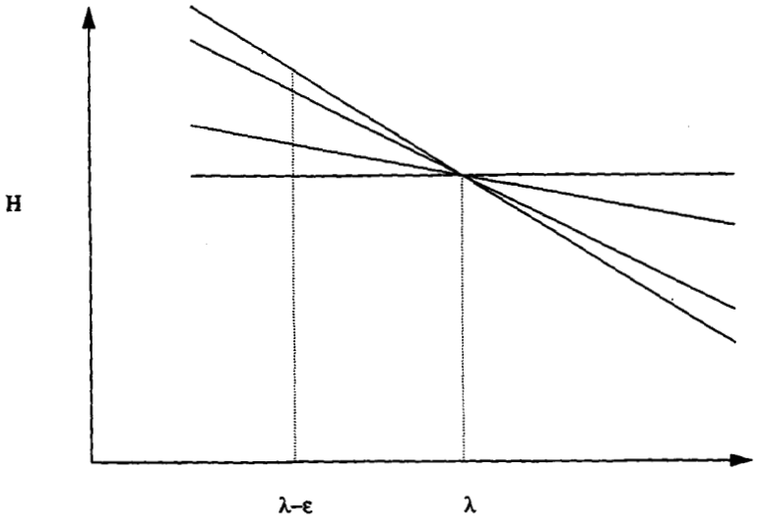

Figure 1. Competing alignments.

with the left and right linear segments of $H(\lambda)$, $\lambda \in [0, \infty]$. If the intersection of these linear functions (x, y) satisfies $H(x)=y$, then the function $H(\lambda)$ is known for all $\lambda \in [0, \infty]$. If not, $H(x)>y$ and $H(x-\varepsilon)$ gives the optimal line that extends $(x, H(x))$ to the left. This line belongs to the final solution $H(\lambda)$, $\lambda \in [0, \infty]$. We recursively apply this method until $H(x)=y$, and we have determined the left-most interval $[0, x]$. In this way, the curve $H(\lambda)$, $\lambda \in [0, \infty]$ can be determined.

In the existing approaches to two-dimensional problems, the one-dimensional algorithm is recursively applied. Given the line through $(\lambda+\varepsilon, H(\lambda+\varepsilon))$ it is necessary to find the next intersection point on the piecewise linear curve $(x, H(x))$. Clearly this can be accomplished by the method described above. However, infinitesimals can speed up this computation. At each (i, j), $H_{i,j}(\lambda+\varepsilon)=u_{ij}+v_{ij}\varepsilon$ is the score for the best line of any alignment ending with x_i and y_j with penalty $\lambda+\varepsilon$. Each of these lines can be seen as competition for $H(\lambda+\varepsilon)=u+v\varepsilon$. Assume $H(\lambda+\varepsilon)=u+v\varepsilon$ with $v<0$. Then if also $v_{ij}>v$:

$$u_{ij}+v_{ij} \geqslant u+vx$$

or

$$x \leqslant \frac{u-u_{ij}}{v_{ij}-v}$$

and the intersection point is at $x=(u-u_{ij})/(v_{ij}-v)$. We are interested in

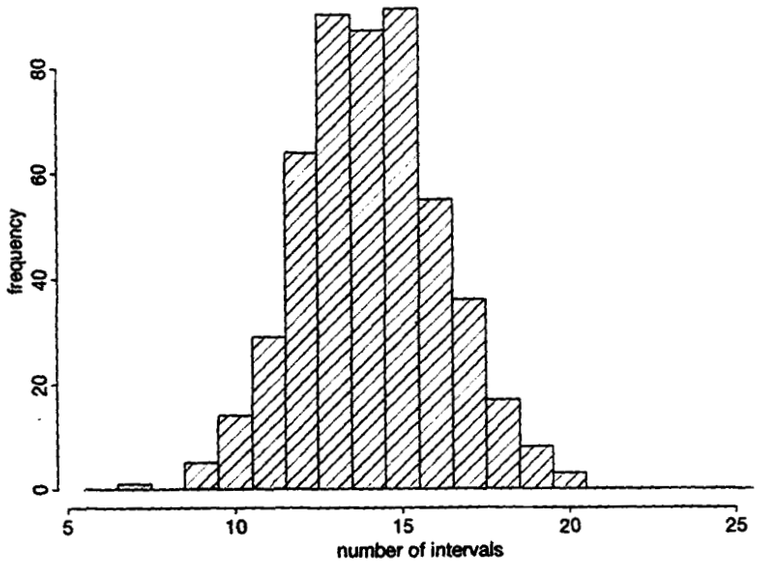

Figure 2. Histogram of the number of intervals.

intersections $\lambda < x < \infty$, and there is the possibility that no such intersections exist. In that case we solve:

$$0 \geqslant u + vx$$

or $x \geqslant -u/v$. Our best estimate of the intersection point is then

$$\varepsilon^* = \min\{-u/v;\ (u-u_{ij})/(v_{ij}-v)\text{: all } (i,j) \text{ where } v_{ij} > v\}.$$

The idea is that we begin the method of the previous paragraph at $(\varepsilon^*, H(\varepsilon^*))$ instead of $(\infty, H(\infty))$. A test of how good this method is is described next.

To illustrate the significant advantages of starting our intersection method at ε^* and not at ∞, we present a simulation as no rigorous bounds have been derived. The simulation is of uniform DNA sequences of length 500. We have set $\mu = \delta = \lambda$ and we use a sample size of 500. Figure 2 shows a histogram of the total number of intervals in $[0, \infty]$. The mean is 14.08 and the standard deviation is 2.11. The ε^* method works least efficiently when we begin at $\lambda = 0$. In Fig. 3 we give the histogram of the number of intervals between $\lambda = 0$ and ε^*. Here the mean is 2.46 and the standard deviation is 2.11. Instead of an average of 14 intervals, now we have only 2.46. Finally, we give a similar histogram for $\lambda = 1.25$ in Fig. 4 where the mean is 1.1 and the standard deviation is 0.32.

2.2.2. Two dimensions. For two-dimensional parameter systems, the first job to extend our method to $S(\mu, \delta) = a - b\mu - c\delta$ is to extend our number

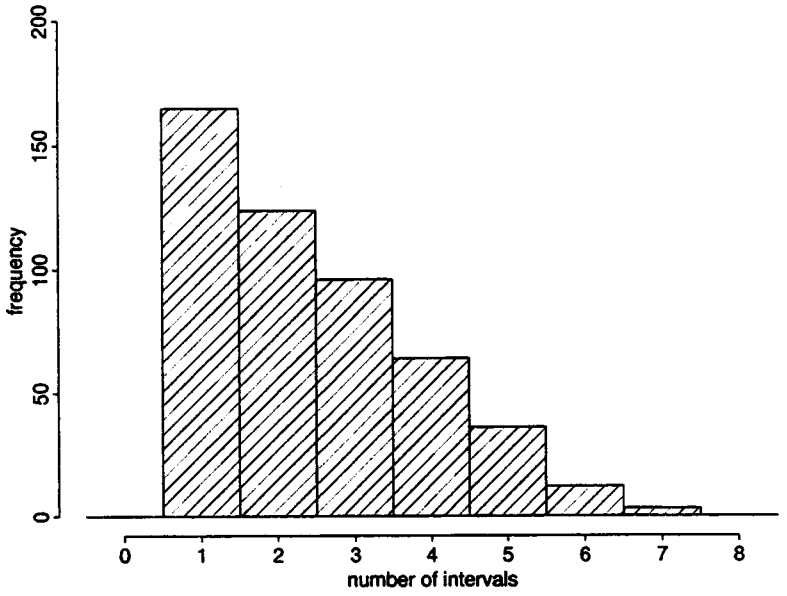

Figure 3. Histogram of the number of intervals between $\lambda+\varepsilon$ and ε^*, for $\lambda=0$.

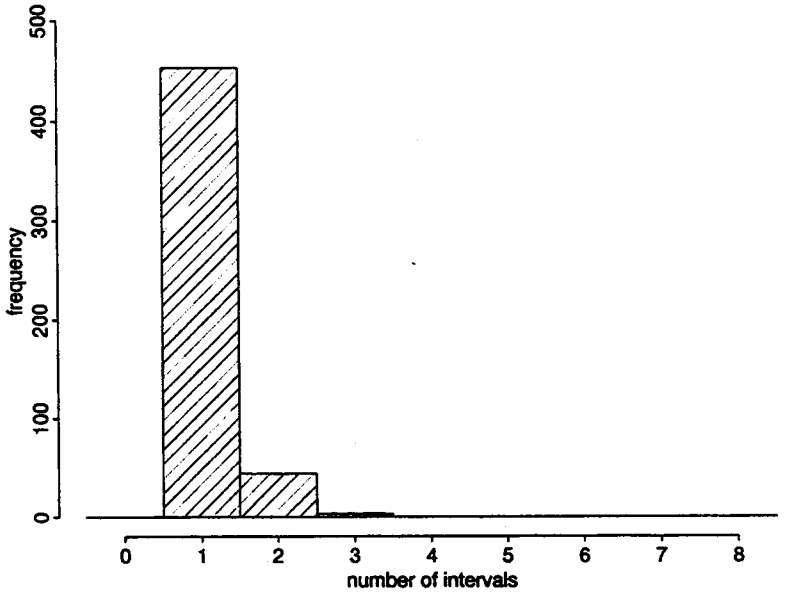

Figure 4. Histogram of the number of intervals between $\lambda+\varepsilon$ and ε^*, for $\lambda=1.25$.

system to two orders of infinitesimals $(\varepsilon_1, \varepsilon_2)$, where each of ε_1 and ε_2 are two-dimensional vectors. Set $x = u_1 + v_1\varepsilon_1 + w_1\varepsilon_2$ and $y = u_2 + v_2\varepsilon_1 + w_2\varepsilon_2$. We have $x > y$ when $u_1 > u_2$; or $u_1 = u_2$ and $v_1 > v_2$; or $u_1 = u_2$, $v_1 = v_2$ and $w_1 > w_2$. As before, any finite multiple of ε_1 is less than any $u > 0$, and any finite multiple of ε_2 is less than ε_1.

The two-dimensional algorithm is based on an extension of (μ, δ) to find the optimal $H(\mu, \delta) = a - b\mu - c\delta$ surface (alignment hyperplane) in the direction (d, e) from (μ, δ). Of course, an infinitesimal distance $\varepsilon_1\sqrt{d^2+e^2}$ in the direction (d, e) might coincide with a line resulting from the intersection of optimal hyperplanes. Therefore we move in the orthogonal direction $(-e, d)$ or $(e, -d)$ a distance $\varepsilon_2\sqrt{d^2+e^2}$. Thus the parameters become $(\mu, \delta) + \varepsilon_1(d, e) + \varepsilon_2(-e, d)$ or $(\mu, \delta) + \varepsilon_1(d, e) + \varepsilon_2(e, -d)$.

The two-dimensional algorithm proceeds by finding convex polygons of constant optimal alignment hyperplanes in $[0, \infty]^2$. Consider $[0, \infty]^2$ as a polygon of 4 edges E and 4 vertices V. Begin at vertex $V_1 = (0, 0)$. Apply the one-dimensional algorithm to edge $E_1 = [(0, 0), (\infty, 0)]$, to find the first optimal interval (V_1, V_2) on E_1. The object is to determine the polygon $V_1, E_1; V_2, E_2; \ldots V_k, E_k; V_{k+1} = V_1$. We agree to proceed counter-clockwise around the polygon as illustrated in Fig. 5. To find E_1, we use the two-dimensional algorithm to determine the alignment hyperplane f adjacent to V_1 in the direction E_1. V_2 is found using the one-dimensional algorithm. Then determine the alignment hyperplane f_1 adjacent to the line E_1 beyond V_2. The intersection $l = f \bigcap f_1$ is a line that has optimal alignment hyperplane f_2 adjacent and counterclockwise. When $f = f_2$, then l determines E_2. Otherwise recursively intersect f and f_2, until the intersection determines E_2. When $V_{k+1} = V_1$, the polygon has been determined. The polygon can then be removed from $[0, \infty]^2$ forming a new region. The procedure of determining a polygon is repeated at a vertex on the boundary of the new region until all convex polygons have been determined.

There is a two-dimensional version of the ε^* method for local alignment to find the nearest intersection point. Referring to Fig. 6, suppose we are at V_1 and wish to find the leftmost line leaving V_2. Instead of operating in one-dimension to find V_2 and then recursively locating the leftmost line leaving V_2, construct a line parallel to (V_1, V_2) an infinitesimal distance ε_1 away. Then beginning from $V_1 + (\varepsilon_1, \varepsilon_2)$ we can use $H_{i,j}$ as a function of ε_2 to find the closest intersection point where the x-coordinate has a larger real part than V_1. Hopefully this will save some of the $f_1, f_2, \ldots$ recursions as shown in Fig. 5.

2.2.3. Three dimensions. Here we briefly sketch a modification of the two-dimensional method to produce three-dimensional tesselations. Obviously our approach will use three infinitesimals and without loss of generality we tesselate $[0, \infty]^3$. For example, our parameters might be $\lambda_1 = \mu$

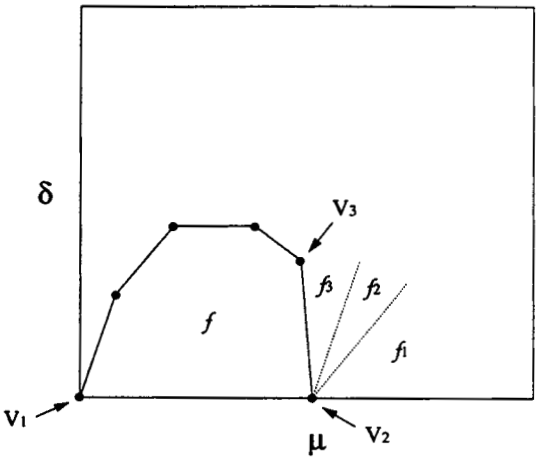

Figure 5. Tesselation in two dimensions.

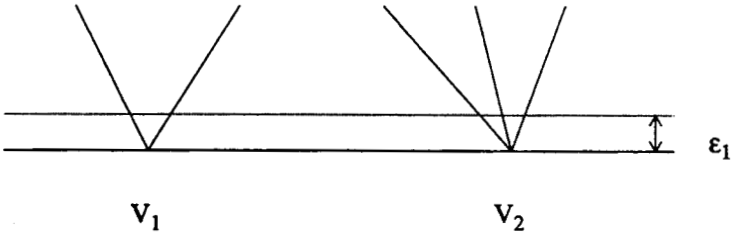

Figure 6. Two-dimensional ε^* method.

and a function $g(k)=\lambda_2+\lambda_3 k$ for the cost of a gap of length k. In Fig. 7, $\mathscr{H}$ and $\mathscr{H}_1$ are optimal hyperplanes associated with $(\mu, 0, 0)-\varepsilon_1(1, 0, 0)+\varepsilon_2(0, 1, 0)+\varepsilon_3(0, 0, 1)$ and $(\mu, 0, 0)+\varepsilon_1(1, 0, 0)+\varepsilon_2(0, 1, 0)+\varepsilon_3(0, 0, 1)$, respectively. The intersection $\mathscr{H}\bigcap\mathscr{H}_1$ results in a plane that can be used to compute an optimal hyperplane $\mathscr{H}_2$. If $\mathscr{H}=\mathscr{H}_2$, we are finished. Otherwise repeat the operation with $\mathscr{H}\bigcap\mathscr{H}_2$. Eventually we reach the optimal plane leaving $(\mu, 0, 0)$ and recursively trace out the planes and edges leaving the (λ_1, λ_2) plane (see Fig. 8). Each of the "unfinished" faces can be closed until the polyhedron is determined.

Our approach to three-dimensional tesselations is to keep a record of the "face" of the remaining portion of $[0, \infty]^3$ beginning with a two-dimensional tesselation of the (λ_1, λ_2) plane. The removed face of the polyhedra is replaced by the interior faces of the polyhedra in the cube. Then any one of these polygons can be a starting point of a further three-dimensional tesselation. The algorithm converges whe the entire cube has been removed, i.e. tesselated.

In Vingron and Waterman (1994) we give a three-dimensional polyhedron

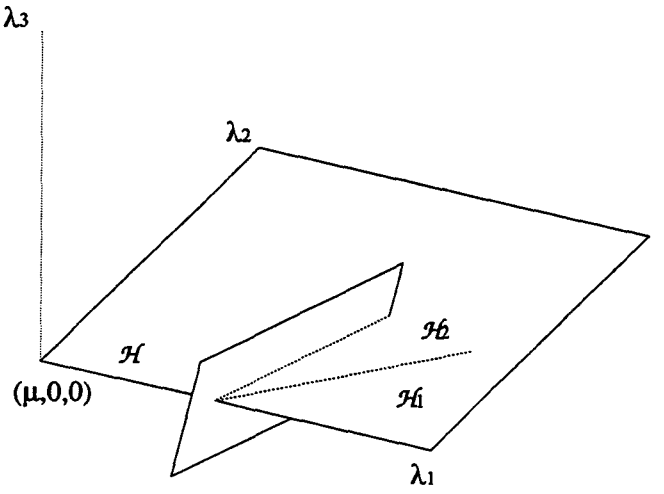

Figure 7. Tesselation in three dimensions.

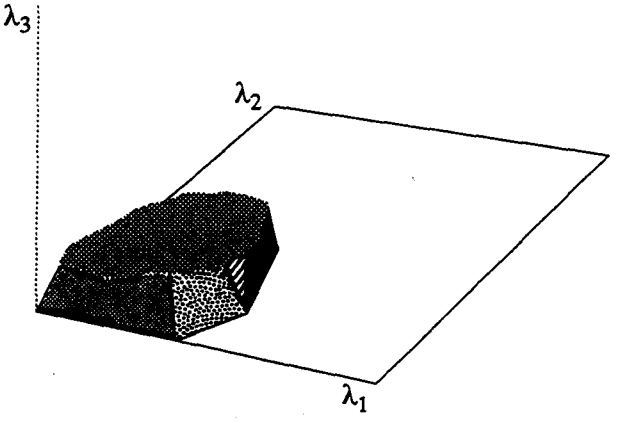

Figure 8. Unfinished three-dimensional tesselation.

for a comparison of two immunoglobins. It was not produced by a three-dimensional algorithm and we hope that someone pursues these issues further. There are many possible approaches and it remains for someone to explore which is most efficient or practical.

3. Ensemble Alignment Hyperplanes. Recall that $\mathscr{A}$ is the ensemble of all alignments. While above we considered $S: \mathscr{A} \rightarrow \mathbf{R}$, here we will consider S to be mapping of $\mathscr{A}$ to the set of alignment hyperplanes. Each alignment can be

mapped to an alignment hyperplane in the following manner. If an alignment has a identities, b mismatches and c indels, the score S is of the form:

$$S=S(\mu, \delta)=a-b\mu-c\delta,$$

which we will refer to as an alignment hyperplane. As above, for global alignment $2a+2b+c=n+m$ and, for local alignment hyperplanes, $2a+2b+c\leqslant n+m$. We are going to study alignment scores as μ and δ vary. We recall that $\max_{\mathscr{A}} S(\mu, \delta)$ is piecewise linear, uniformly continuous, and concave up. This surface can be computed by the parametric alignment algorithms in Section 2-2.

In this section we will describe simple extensions of the dynamic programming algorithms for optimal alignments that will compute $\mathscr{S}=S(\mathscr{A})$, the set of optimal hyperplanes and their frequencies. Due to the huge number of hyperplanes, we will derive in Section 3.3 a more efficient algorithm to find all hyperplanes within d of the optimal scores $\max\{S(\mu, \delta): \mathscr{A}\}$.

3.1. *Global alignment hyperplanes.* The setup is analogous to that for global optimal alignment. Define:

$$\mathscr{S}_{i,j}=\{S(\mu, \delta): S(\mu, \delta) \text{ is an alignment hyperplane for } x_1, \ldots, x_i \text{ and } y_1, \ldots, y_j\}.$$

Set $\mathscr{S}_{0,0}=\phi$, $\mathscr{S}_{i,0}=\{-i\delta\}$, $\mathscr{S}_{0,j}=\{-j\delta\}$ for $1\leqslant i\leqslant n$, $1\leqslant j\leqslant m$.
Define:

$$\mathscr{S}+c\equiv \bigcup_{S\in\mathscr{S}} \{S+c\}.$$

Then the recursion for $\mathscr{S}_{i,j}$ is:

$$\mathscr{S}_{i,j}=[\mathscr{S}_{i-1,j-1}+s(x_i, y_j)]\bigcup[\mathscr{S}_{i-1,j}+(-\delta)]\bigcup[\mathscr{S}_{i,j-1}+(-\delta)]. \quad (3)$$

The proof of this recursion follows the usual reasoning for dynamic programming algorithms for sequence alignment. An alignment ending at (i, j) can terminate in one of three ways:

$$\begin{array}{ccc} \ldots x_i & \ldots x_i & \ldots - \\ \ldots y_j & \ldots - & \ldots y_j. \end{array}$$

These correspond, respectively, to the terms:

$$\mathscr{S}_{i-1,j-1}+s(x_i, y_j) \qquad \mathscr{S}_{i-1,j}+(-\delta) \qquad \mathscr{S}_{i,j-1}+(-\delta)$$

of equation (3), the union of which comprises $\mathscr{S}_{i,j}$.

There is a bit more work to describe an algorithm to compute $\mathcal{S}_{i,j}$. The three lists of equation (3) must be merged. Assume $\mathcal{S}_{i,j}$ is ordered lexicographically on (a, b, c). Then merging of the three ordered lists in equation (3) to form $\mathcal{S}_{i,j}$ can be done in time proportional to the size of the list $\mathcal{S}_{i,j}$. How large is $|\mathcal{S}_{i,j}|$? Set $n(a, b, c)$ to be the number of alignments A with score $a-b\mu-c\delta$.

LEMMA 2. *(1)* $|\mathcal{S}_{i,j}| = O((min(i, j))^2) = O(ij)$
(2) $\sum_{(a,b,c)} n(a, b, c) = O(\binom{i+j}{i})$.

Proof. For $f(\mu, \delta) = a-b\mu-c\delta \in \mathcal{S}_{i,j}$, we have $2(a+b)+c = i+j$ where $a, b, c \in \{0, 1, 2, \ldots\}$. Therefore the result follows. Part (2) is simply a restatement of the fact that there are $\binom{i+j}{i}$ alignments in total $n(a, b, c)$ having form $f(\mu, \delta) = a-b\mu-c\delta$. ■

Proposition 1. The running time of the above global hyperplane algorithm is $O(n^4)$.

To compute the frequencies of $f(\mu, \delta)$, extend the elements $f(\mu, \delta)$ of $\mathcal{S}$ to $(n, f(\mu, \delta))$. The boundary conditions are $\mathcal{S}_{0,0} = \phi$, $\mathcal{S}_{i,0} = \{(1, -i\delta)\}$, $\mathcal{S}_{0,j} = \{(1, -j\delta)\}$. With this new definition, we set:

$$\mathcal{S}+c = \bigcup_{(n,S)\in\mathcal{S}} \{(n, S+c)\}.$$

It is of interest to know the number of alignments with alignment hyperplane $a-b\mu-c\delta$. While there is sometimes a unique alignment with a given hyperplane, often there are many such alignments. The algorithm for computing the hyperplanes is equation (1) and the algorithm for computing the frequencies is as follows.

Suppose $a-b\mu-c\delta$ is in $\mathcal{S}_{i,j}$ and define:

$n_{i-1,j}$ to be the frequency of $a-b\mu-(c-1)\delta$ in $\mathcal{S}_{i-1,j}$

$n_{i-1,j-1}$ to be the frequency of $(a-1)-b\mu-c\delta$ in $\mathcal{S}_{i-1,j-1}$ if $a_i = b_j$
and of $a-(b-1)\mu-c\delta$ in $S_{i-1,j-1}$ if $a_i \neq b_j$.

$n_{i,j-1}$ to be the frequency of $a-b\mu-(c-1)\delta$ in $\mathcal{S}_{i,j-1}$.

Then the frequency n of $a-b\mu-c\delta$ in $\mathcal{S}_{i,j}$ is:

$$n \equiv n(a, b, c) \equiv n_{i,j} = n_{i-1,j} + n_{i-1,j-1} + n_{i,j-1}$$

and we have

$$(n, a-b\mu-c\delta) \in \mathcal{S}_{i,j}.$$

We illustrate this algorithm in Fig. 9 with a simple example. These

	C	T	G	T	C
T	2 0 0 2 1 0 1 0	3 0 0 3 1 0 1 1 1 1 0 1	4 0 0 4 2 0 1 2 1 1 0 2	5 0 0 5 2 0 1 3 2 1 0 3	6 0 0 6 3 0 1 4 2 1 0 4
G	3 0 0 3 2 0 1 1	6 0 0 4 5 0 1 2 1 0 2 0 1 1 0 2	10 0 0 5 7 0 1 3 1 0 2 1 5 1 0 3 1 1 1 1 1 2 0 1	15 0 0 6 13 0 1 4 3 0 2 2 7 1 0 4 2 1 1 2 1 2 0 2	21 0 0 7 21 0 1 5 5 0 2 3 9 1 0 5 4 1 1 3 1 2 0 3
C	4 0 0 4 2 0 1 2 1 1 0 2	10 0 0 5 10 0 1 3 3 0 2 1 2 1 0 3	20 0 0 6 23 0 1 4 9 0 2 2 1 0 3 0 7 1 0 4 2 1 1 2 1 2 0 2	35 0 0 7 46 0 1 5 19 0 2 3 2 0 3 1 14 1 0 5 9 1 1 3 1 1 2 1 2 2 0 3 1 2 1 1	56 0 0 8 67 0 1 6 24 0 2 4 2 0 3 2 38 1 0 6 26 1 1 4 4 1 2 2 10 2 0 4 3 2 1 2 1 3 0 2
C	5 0 0 5 2 0 1 3 2 1 0 3	15 0 0 6 16 0 1 4 5 0 2 2 4 1 0 4 4 1 1 2	35 0 0 7 49 0 1 5 24 0 2 3 4 0 3 1 11 1 0 5 5 1 1 3 1 2 0 3	70 0 0 8 115 0 1 6 66 0 2 4 15 0 3 2 1 0 4 0 25 1 0 6 21 1 1 4 3 1 2 2 3 2 0 4 2 2 1 2	126 0 0 9 182 0 1 7 90 0 2 5 17 0 3 3 1 0 4 1 98 1 0 7 93 1 1 5 26 1 2 3 2 1 3 1 27 2 0 5 14 2 1 3 1 2 2 1 3 3 0 3 1 3 1 1

Figure 9. Global alignment: $(n(a, b, c), a, b, c)$.

combinatories count order of deletions. For example $\mathscr{S}_{1,2} = (3, -3\delta)$ which appears as 3 0 0 3 in the table has the three alignments:

$$\begin{array}{ccc} C\,T- & C-T & -C\,T \\ --T & -T- & T-- \end{array}$$

3.2. *Local alignment hyperplanes.* Now we develop the corresponding ideas for local optimal alignments. Due to the nature of local alignmenta, we define a local alignment hyperplane for $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ to be an alignment hyperplane for $x_k x_{k+1} \ldots x_i$ and $y_l y_{l+1} \ldots y_j$; $1 \leqslant k \leqslant i \leqslant n$, and $1 \leqslant l \leqslant j \leqslant m$:

where $\begin{matrix} x_k \\ y_l \end{matrix}$ and $\begin{matrix} x_i \\ y_j \end{matrix}$

are both identities. This definition is motivated by the fact that if local

alignments do not begin and end with identities, the score can be trivially improved for all $(\mu, \delta) > (0, 0)$ by trimming back to the identities nearest the beginning and end of the alignments or to $\emptyset$ if no identities exist. The definition of $\mathcal{S}_{ij}$ is suitably modified:

$S^*_{i,j} = \{S(\mu, \delta) : S(\mu, \delta)$ is a local alignment hyperplane ending at $(x_i, y_j)\}$.

Define:

$$\mathcal{S} + c = \begin{cases} \bigcup_{S \in \mathcal{S}} \{S + c\}, & \mathcal{S} \neq \emptyset \\ \emptyset, & \mathcal{S} = \emptyset. \end{cases}$$

To begin the recursion set $\mathcal{S}_{i,j} = \mathcal{S}^*_{i,j} = \emptyset$ if $i \cdot j = 0$:

$$\mathcal{S}_{i,j} = [\mathcal{S}_{i-1,j-1} + s(x_i, y_j)] \bigcup [\mathcal{S}_{i-1,j} - \delta] \bigcup [\mathcal{S}_{i,j-1} - \delta] \bigcup \{s(x_i, y_j)\} I\{x_i = y_j\}$$

$$\mathcal{S}^*_{i,j} = \begin{cases} \{[\mathcal{S}_{i-1,j-1} + s(x_i, y_j)] \bigcup \{s(x_i, y_j)\}\} & \text{if } x_i = y_j \\ \emptyset, & \text{if } x_i \neq y_j. \end{cases} \tag{4}$$

This recursion is of course based on the algorithm for optimal local alignment. The "0" of equation (4) above has disappeared however. Whenever $x_i = y_j$ we begin a new alignment, regardless of the sequences $x_1 \ldots x_{i-1}$ and $y_1 \ldots y_{j-1}$. Otherwise the logic is straightforward. $\mathcal{S}_{i,j}$ collects all previous alignments ending them with the aligned pair (x_i, y_i) or with a deletion. The convention $\emptyset$ excludes alignments beginning with deletions or mismatches. $\mathcal{S}^*_{i,j}$ simply collects the subset of $\mathcal{S}_{i,j}$ ending with an identity (x_i, y_j) where $x_i = y_j$.

As with the global hyperplane algorithm ordered lists must be merged, in time proportional to list length.

Proposition 2. (1) $|\mathcal{S}_{ij}| = O(\max\{i, j\}\,(\min\{i, j\})^2)$.

(2) *The running time of the local hyperplane algorithm is* $O(n^5)$.

Proof. For $f(\mu, \delta) = a - b\mu - c\delta \in \mathcal{S}_{i,j}$, we have $2(a+b) + c = i' + j'$ where $a, b, c \in \{0, 1, 2, \ldots\}$ and $0 \leqslant i' \leqslant i, 0 \leqslant j' \leqslant j$. Both a and b are restricted between 0 and $\min\{i, j\}$ while c can vary between 0 and $\max\{i, j\}$. Of course $|\mathcal{S}_{n,m}| = O(n^3)$ which implies an $O(n^5)$ algorithm. ■

A recursive procedure corresponding to the local hyperplane algorithm gives the frequencies of the hyperplanes. The algorithm is illustrated in Fig. 10 by the same simple example we used for global hyperplanes.

3.3. *Near optimal hyperplanes.* Computing all alignment hyperplanes is made difficult by the sheer number of hyperplanes ($O(n^2)$ for global, $O(n^3)$ for local) as well as the $O(n^2)$ factor for all (i, j) pairs. The motivation of this work was to study the structure of the optimal hyperplanes and those hyperplanes

	C	T	G	T	C
T		1 0 0 1		1 1 0 0	
G			2 1 0 0 1 2 0 0		
C	1 1 0 0				5 1 0 0 1 2 0 0 2 2 0 1 3 2 0 3 1 2 1 1 1 3 0 1
C	2 1 0 0				7 1 0 0 1 2 0 0 2 2 0 1 3 2 0 2 4 2 0 3 6 2 0 4 1 2 1 0 1 2 1 1 4 2 1 2 2 3 0 2 1 3 1 0

Figure 10. Local alignment: ($n(a, b, c)$, a, b, c).

close to optimal. Consequently we now consider the problem of computing all hyperplanes that come within d of the optimal, for $d \geq 0$. Obviously for $d = 0$, we compute only the optimal hyperplane.

Let $\mathcal{S}_{i,j}$ be the set of alignment hyperplanes for $x_1, \ldots x_i$ and $y_1, \ldots y_j$. Our discussion holds for both the local and global cases. Define:

$$M_{i,j}(\mu, \delta) = \max\{a - b\mu - c\delta = f(\mu, \delta) : f(\mu, \delta) \in \mathcal{S}_{i,j}\}.$$

Now consider all hyperplanes that come within d of $M_{i,j}$ at some μ and δ defined by:

$$O_{i,j} = \left\{a - b\mu - c\delta = f(\mu, \delta) \in \mathcal{S}_{i,j} : \min_{(\mu,\delta)} |M_{i,j}(\mu, \delta) - f(\mu, \delta)| \leq d\right\}.$$

We now observe that recursion with $O_{i,j}$ instead of $\mathcal{S}_{i,j}$ will not change the

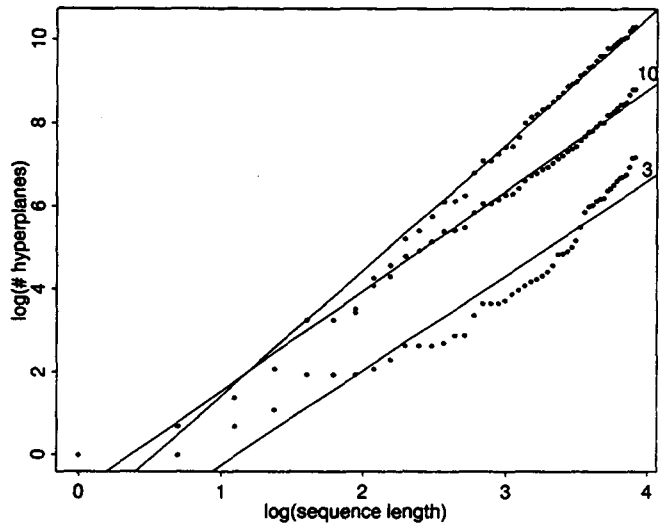

Figure 11. Growth of the number of local hyperplanes as a function of $d=\infty$ (unlabeled), $d=10$, and $d=3$.

final set $O_{n,m}$ of near optimal hyperplanes. This is because all $f(\mu, \delta) \in \mathscr{S}_{i,j} \bigcap O_{i,j}$ are of distance greater than d from $M_{i,j}$. If $f(\mu, \delta)$ contributes to $O_{n,m}$ then we write the full alignment score by:

$$f(\mu, \delta)+g(\mu, \delta) \in O_{n,m},$$

where $g(\mu, \delta)$ is the contribution from $x_{i+1} \ldots x_n$, $y_{j+1} \ldots y_m$. Then:

$$M_{n,m}(\mu, \delta)-(f(\mu, \delta)+g(\mu, \delta))=(M_{i,j}-f(\mu, \delta)) + \{(M_{n,m}-M_{i,j})-g(\mu, \delta)\} \geqslant M_{i,j}-f(\mu, \delta)>d.$$

The last inequality holds since $M_{n,m}-M_{i,j}$ is at least as large as the best score of aligning $x_{i+1} \ldots x_n$ with $y_{j+1} \ldots y_m$ and $g(\mu, \delta)$ cannot be larger than that score.

It is expensive to produce $O_{i,j}$ since M_{ij} must be calculated. The above idea can be utilized as follows. Compute $\mathscr{S}_{i,j}$ row by row stopping when $|\mathscr{S}_{i,j}|=N$, some suitably large number. If we stop at (i, j_1) and $(i+1, j_{i+1})$ with $j_i > j_{i+1}$, add $(i, j_{i+1}+1)$ $(i, j_{i+1}+2) \ldots (i, j_i-1)$ to make a "boundary" across the matrix, from the top (row 1) to the bottom (row n). Then replace $S_{i,j}$ by $O_{i,j}$ on the boundary so that the succeeding recursions use the smaller sets. This significantly reduces storage and running time.

As our interest is principally local alignments we have implemented the truncation method for the local alignment algorithm. We performed simulations shown in Figs 11, 12 to get an indication of the growth of the number of alignment hyperplanes and the number of alignments as a function of the

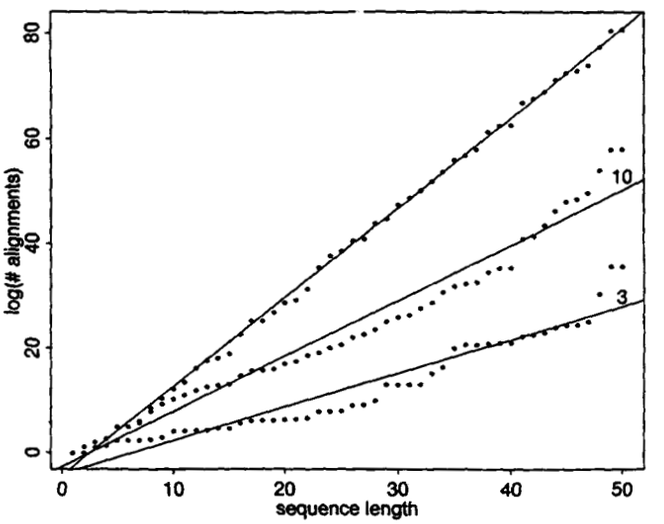

Figure 12. Growth of the number of global hyperplanes as a function of $d=\infty$ (unlabeled), $d=10$, and $d=3$.

truncation parameter. The sequences of length up to 50 of uniformly distributed letters were generated by successively adding letters to sequences already analysed. (This gave us the advantage of using the computations from the previous data point and gave the disadvantage of serial dependencies among the data points.) Recall that the number of global [local] hyperplanes should grow quadratically $O(n^2)[O(n^3)]$ with sequence length n while the number of alignments should grow exponentially. The lines for $d=\infty$ reflect this feature. While truncation does not affect the polynomial/exponential properties, it obviously greatly changes the coefficient of the polynomial or exponential.

3.4. *More examples.* While in parametric optimal local alignment, the surface $(\mu, \delta, H(\mu, \delta))$ can be meaningfully displayed, this is not the case for parametric local alignment. Hyperplanes intersect and overlay one another in complex ways, and we can display sections $(\mu(\lambda), \delta(\lambda), H(\lambda))$ where μ and δ are linear functions of λ. For a simple example, take x=TGCCGTG and y=CTGTCGCTGCACG. Two optimal global alignments exist:

$$\begin{array}{c} -\,T\,G\,C\,C\,G-T\,G \\ C\,T\,G\,T\,C\,G-T\,G\,A\,C\,G \end{array}$$

and

$$\begin{array}{c} -\,T\,G\,C-C\,G-T\,G \\ C\,T\,G-T\,C\,G\,C\,T\,G\,A\,C\,G. \end{array}$$

The first alignment is optimal for $\delta \geqslant 2\mu$ and the second for $\delta \leqslant 2\mu$. Figure 13

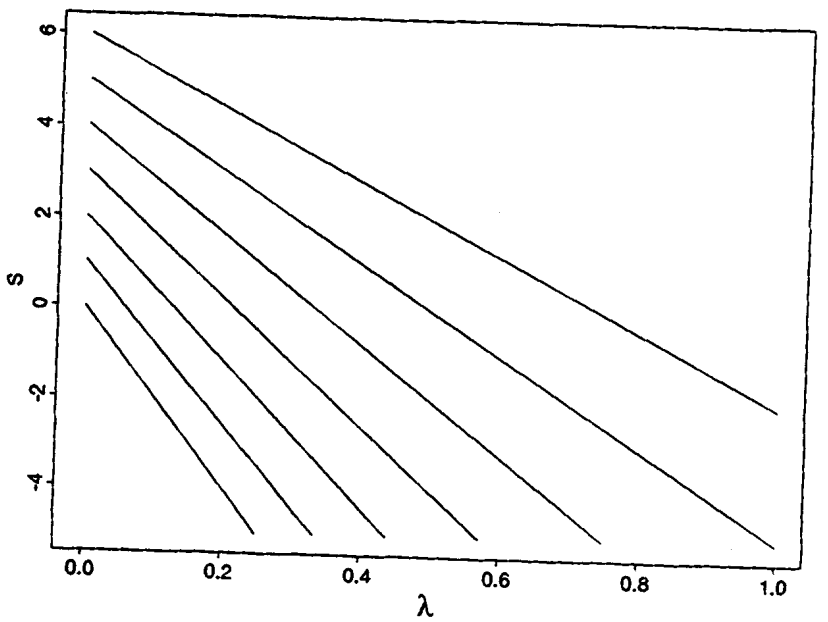

Figure 13. Global hyperplanes for $\mu=2\lambda$, $\delta=\lambda$.

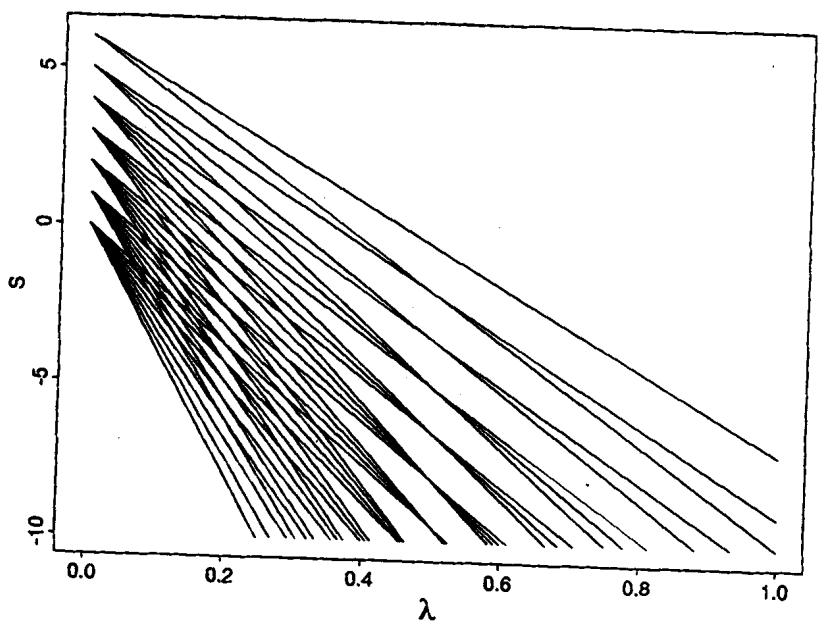

Figure 14. Global hyperplanes for $\mu=\lambda$, $\delta=2\lambda$.

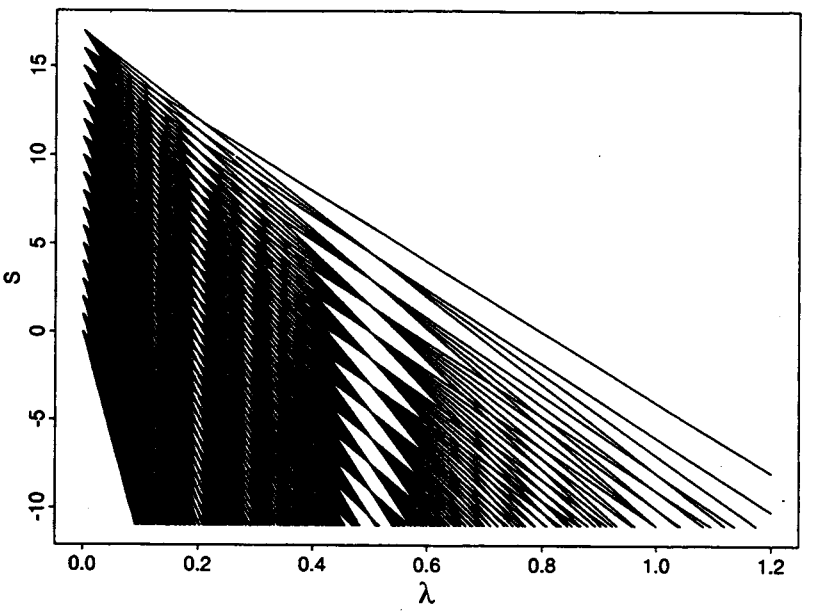

Figure 15. Global hyperplanes at $\mu=\lambda$ and $\delta=2\lambda$.

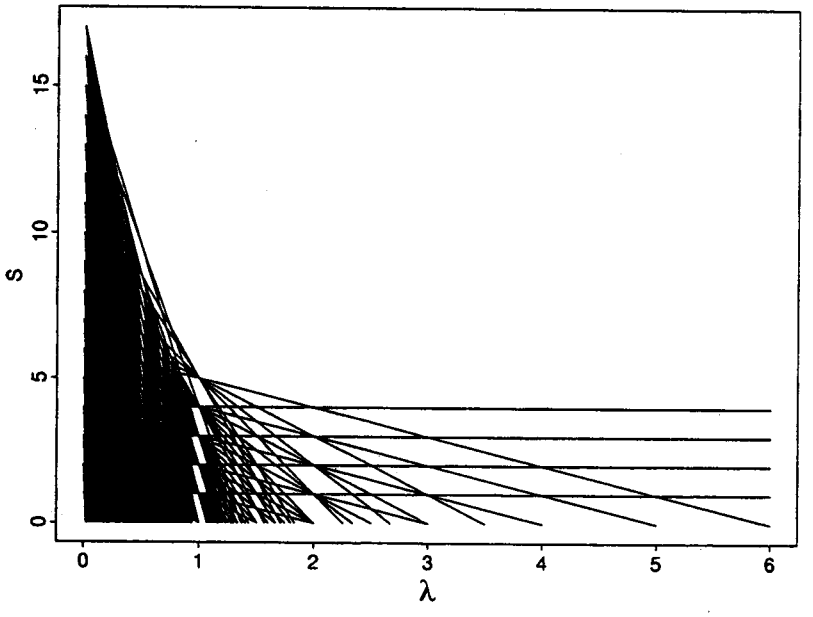

Figure 16. Local hyperplanes at $\mu=\lambda$ and $\delta=2\lambda$.

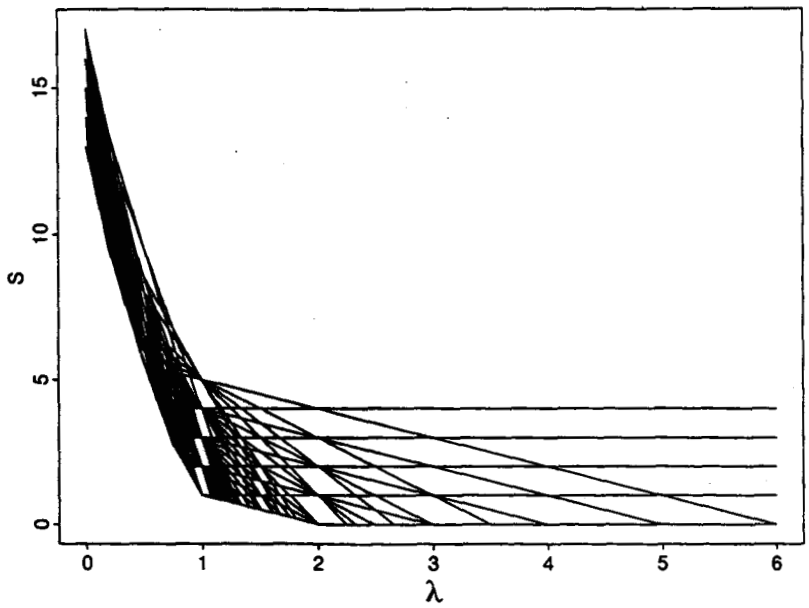

Figure 17. Local hyperplanes at $\mu=\lambda$ and $\delta=2\lambda$.

shows the global hyperplanes for $\mu=2\lambda, \delta=\lambda$. Figure 14 shows the hyperplanes for $\mu=\lambda, \delta=2\lambda$. The pencil of lines (i.e. hyperplanes) emanating from the 7 points in Fig. 14 are coincident into 7 lines in Fig. 13. This occurs because in global alignments the number of letters not involved in identities is equal for all alignments with the same number of identities, i.e. for each pencil.

Next we present an analysis of two random DNA sequences of length $n=30$. A global alignment section is shown in Fig. 15 for $\mu=\lambda$ and $\delta=2\mu$. Here there are 370 planes and a total of 9.6×10^{21} alignments. In Fig. 16, the local alignments for the same section, $\mu=\lambda$ and $\delta=2\lambda$ are presented. There are 5144 local hyperplanes with 6.4×10^{18} alignments. Recall that local alignments must begin and end with identities (matches). This accounts for the fact that there are fewer local alignments than global alignments reported. Finally, we truncate the local alignments at distance $d=4$ from the optimal. In Fig. 17 there are 720 hyperplanes with 1.1×10^{12} alignments.

The author thanks M. Eggert and S. Xu for the programming assistance and the referee for many useful comments. This research was supported by grants from the National Institutes of Health and the National Science Foundation.

LITERATURE

Byers, T. H. and M. S. Waterman. 1984. Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming. *Oper. Res.* **32**, 1381.

Fernandez-Baca, D. and S. Srinivasan. 1991. Constructing the minimization diagram of a two-parameter problem. *Oper. Res. Lett.* **10**, 87–93.

Finkelstein, A. V. and M. A. Roytberg. 1993. Computation of biopolymers: a general approach to different problems. *BioSystems* **30**, 1–19.

Gusfield, D., K. Balasubramian and D. Naor. 1992. In *Proceedings of the Third Annual ACM-SIAM Discrete Algorithms*, 432–439.

Howell, J. A., T. F. Smith and M. S. Waterman. 1980. Computation of generating functions for biological molecules. *SIAM J. Appl Math.* **39**, 119–133.

McCaskill, J. S. 1990. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers* **29**, 1105–1119.

Naor, D. and D. Brutlag. 1993. On Suboptimal alignments of biological sequences. *Proc. of the Fourth International Symposium on Combinatorial Pattern Matching*, Padova, Italy, June 1993. Lecture Notes in Computer Science.

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. mol. Biol.* **48**, 443–453.

Tavaré, S. 1986. Some mathematical questions in biology—DNA sequence analysis. In *Lectures on Mathematics in the Life Sciences*, **17**, 29–56. The American Mathematical Society, Providence, Rhode Island.

Thorne, J. L., H. Kishino and J. Felsenstein. 1991. An evolutionary model for maximum likelihood alignment of DNA sequences. *J. mol. Evol.* **33**, 114–124.

Thorne, J. L., H. Kishino and J. Felsenstein. 1992. Inching toward reality: an improved likelihood model of sequence evolution. *J. mol. Evol.* **34**, 3–16.

Vingron, M. and P. Argos. 1990. Determination of reliable regions in protein sequence alignments. *Prot. Engng* **3**, 565–569.

Vingron, M. and M. S. Waterman. 1993. Rapid and accurate estimates of statistical significance for sequence database searches. *J. mol. Biol.* (in press).

Waterman, M. S. 1983. Sequence alignment in the neighbourhood of the optimum with general applications to dynamic programming. *Natl. Acad. Sci. USA* **80**, 3123.

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

Waterman, M. S. 1989. Sequence alignments. In: *Mathematical Methods for DNA Sequences*, M. S. Waterman (Ed.). CRC Press.

Waterman, M. S., M. Eggert and E. Lander. 1992. Parametric sequence comparisons. In *Natl. Acad. Sci. USA*, **89**, 6090–6093.

Zhang, M. Q. and T. G. Marr. 1993. Alignment of molecular sequences by random path analysis. Manuscript.