Algorithmica (1994) 12: 312–326

Algorithmica
© 1994 Springer-Verlag New York Inc.

Parametric Optimization of Sequence Alignment[1]

D. Gusfield,[2] K. Balasubramanian,[2] and D. Naor[2]

Abstract. The *optimal alignment* or the *weighted minimum edit distance* between two DNA or amino acid sequences for a given set of weights is computed by classical dynamic programming techniques, and is widely used in molecular biology. However, in DNA and amino acid sequences there is considerable disagreement about how to weight matches, mismatches, insertions/deletions (indels or spaces), and gaps. *Parametric sequence alignment* is the problem of computing the optimal-valued alignment between two sequences as a *function* of variable weights for matches, mismatches, spaces, and gaps. The goal is to partition the parameter space into regions (which are necessarily convex) such that in each region one alignment is optimal throughout and such that the regions are maximal for this property. In this paper we are primarily concerned with the structure of this convex decomposition, and secondarily with the complexity of computing the decomposition. The most striking results are the following: For the special case where only matches, mismatches, and spaces are counted, and where spaces are counted throughout the alignment, we show that the decomposition is surprisingly simple: all regions are infinite; there are at most $n^{2/3}$ regions; the lines that bound the regions are all of the form $\beta = c + (c + 0.5)\alpha$; and the entire decomposition can be found in $O(knm)$ time, where k is the actual number of regions, and $n < m$ are the lengths of the two strings. These results were found while implementing a large software package for parametric sequence analysis, and in turn have led to faster algorithms for those tasks. A conference version of this paper first appeared in [10].

Key Words. Sequence alignment, Parametric analysis, Edit distance, Sequence homology, Global alignment, Local alignment.

1. Introduction.

Finding the minimum cost edit distance, or the best alignment, of two DNA, RNA, or amino acid sequences has become almost the standard technique for sequence comparison in molecular biology. It is used to determine whether and where two sequences are similar (homologous), to determine evolutionary history between species, to find consensus sequences, and other significant functions. There are literally hundreds of papers written on this topic and its applications to biology. For an introduction and small reflection of this literature see [2], [3], [5]–[7], [12], [13], [18], and [19].

However, in all the present methods for optimal sequence alignment, specific substitution, insertion/deletion (indel), and gap penalties must be specified, and the (biological) significance of the alignment depends heavily upon the "right" choice of the weights. There is considerable disagreement among molecular biologists about the correct choice, and it is probably the case that there is no unique choice for the parameters (as pointed out in [5] and [1] with respect to

[1] This research was partially supported by Grant DE-FG03-90ER60999 from the Department of Energy, and Grants CCR-8803704 and CCR-9103937 from the National Science Foundation.
[2] Computer Science Department, University of California, Davis, CA 95616, USA.

Received August 21, 1991; revised August 15, 1992. Communicated by Alberto Apostolico.

gap penalties). The significance of an alignment is based either on biological grounds, or on its sensitivity to the choice of parameters. Instead of repeatedly varying the parameter weights and solving for the optimal alignment, other parametric methods ought to be employed. As an example, D. Sankoff and J. Kruskal [13, pp. 290–293] demonstrate the difficulties in finding the relative weights for gaps/substitutions and other operations by a specific example: the comparison of human and *E. Coli* 5S RNA (two sequences of 120 characters each over a four-letter alphabet). Their solution involves varying one parameter, the number of *indels*, until its appropriate value is found. Similarly, the paper by Fitch and Smith [5] demonstrates how the biologically accepted alignment may easily be missed if inappropriate weights are used.

There are two ways around the problem of choosing a "correct" choice of parameters. The first is to compute, for a given set of initial guesses for parameter values, the optimal alignment $\mathscr{A}$ at that point and, in addition, a maximal region P such that $\mathscr{A}$ is optimal throughout P. The other, more general, approach is to find the entire decomposition of the parameter space into such maximal regions. Papers [5] and [6] present examples of decompositions that were computed by hand. During the process of developing and analyzing a computer program which is based on these approaches, we were able to characterize this decomposition in several cases, and this considerably simplified and accelerated the algorithm for finding the decomposition.

We formally define the *parametric alignment* problem in Section 1.1 and summarize our results in Section 1.2. In Sections 2 and 3 we consider the two-parameter case, where only substitution and indel weights can vary. Two different variants, the global and the local alignment, are studied. Section 4 deals with the three-parameter case. Finally, in Section 5 we discuss a richer variant of the problem, where independent weights for each type of mismatch are specified.

1.1. Definitions. The *edit distance* between two sequences is the minimum weighted sequence of *edit operations* (insertion, deletion, and substitution of single characters) that must be performed to transform one sequence into another. This has found widespread use as a measure of sequence similarity. It is often more important to know what the actual edit operations are rather than just the total cost or value. These operations can be represented as an *alignment*, and it is often the alignment that is sought.

An *alignment* of two sequences S_1 and S_2 of lengths n and m $(\geq n)$, respectively, is obtained by introducing spaces into the two sequences such that the lengths of the two resulting sequences are identical, and placing these two resultant sequences one upon the other subject to the constraint that no column contains two spaces. Any column that contains two identical characters is called a *match*. Any column that contains two dissimilar characters is called a *mismatch* and any column that contains a space is referred to as a *space* or *indel*. The correspondence between them and the edit operations is straightforward: a mismatch represents a substitution, a space is either an insertion or a deletion, depending whether it is introduced in the source string or the target string, and a match is an untouched character.

A series of one or more contiguous space characters in the same sequence is referred to as *gap*.

An alignment $\mathscr{A}$ may therefore be characterized by the number of matches, mismatches, spaces, and gaps. We denote these quantities by $w_{\mathscr{A}}$, $x_{\mathscr{A}}$, $y_{\mathscr{A}}$, $z_{\mathscr{A}}$, respectively (or w, x, y, z when referring to an unspecified alginment). Note, however, that this representation is many-to-one: different alignments could have the same 4-tupe (w, x, y, z). If α_0 is the mismatch penalty, β_0 is the space (indel) penalty, and γ_0 is the gap penalty, then the *value* of an alignment is defined to be

$$v = w - \alpha_0 x - \beta_0 y - \gamma_0 z.$$

The region of interest is the region where α, β, γ are all positive. We ignore the case where the weight of the matches is also a parameter since we can divide all the parameters by this value and reduce it to the above case without changing the relative order of the value of the alignments. For fixed weights, the value of the optimal (maximum value) alignment of two strings can be found by dynamic programming in $O(nm)$ time [14], a fact that was discovered many times independently.

A given choice of parameter values α_0, β_0, and γ_0 defines an optimal alignment (not necessarily unique). Since an alignment is essentially a discrete object any alignment that is optimal for some fixed values $(\alpha_0, \beta_0, \gamma_0)$ is optimal in a certain *region* in the (α, β, γ) space. Hence, the three-parameter space is decomposed into regions which we call *optimal regions* such that in every region one alignment is optimal throughout and the regions are maximal for this property. This decomposition is completely defined by the two sequences.

The value of an alignment is always a linear function of the parameters; hence it can be easily observed that the regions, which are bounded by the intersection of hyperplanes, are all convex polygons.

The above definition of an alignment is called *global* since it finds the edit distance between the entire two strings, and charges or penalizes for characters throughout the alignment. Another variant of the alignment problem is the *local alignment* problem, and is formally defined below.

The Local Alignment Problem. Given two strings A and B, find substrings A' and B' of A and B, respectively, such that the optimal (global) alignment value of A' and B' is maximum over all pairs of substrings from A and B.

The parametric version of the local alignment problem is to parametrize the (global) alignment value of A' and B'. While local alignment seems more complex than global alignment, when all the parameters are fixed an optimal local alignment can be found in $O(nm)$ time [16], the same bound as for global alignment.

In biological applications, *global* alignments are used for searching for homologies between two sequences, that is, when the whole sequences are expected to be homologous. This is true more often of proteins than in the study of DNA, though global alignment problems also arise as subtasks of more complex alignment problems [14]. In contrast, *local* alignments are used in cases where the two given sequences may not be highly homologous in their entirety, but contain *substrings* that are highly similar.

There is yet another variant of the alignment problem, which is a hybrid of the global and the local. This is the *end-free* alignment. When computing the value of an *end-free* alignment, we ignore any (contiguous) spaces that overhang at the *extreme ends* of the alignment, i.e., we are permitted to delete one suffix and one prefix (perhaps of the same sequence) with no cost. Note that local alignment is a more restrictive version of the *end-free* alignment case. In this paper we consider the two main variants, the global and the local; however, all of our results for the local alignment carry over to the *end-free* case.

1.2. Summary of Results. In this paper we are concerned with the characteristics of the parametric space decomposition, and its algorithmic implications. We are also interested in questions of the type: Given an arbitrary line in the parametric space, how many different regions can it go through? The latter was motivated by our computer program PARAL which is based on a primitive operation that, when given a line in the space, can find all the regions it crosses in time $O(knm)$, where k is the actual number of regions it goes through (using the algorithm of Eisner–Severance [4], [17]; this is essentially Newton's method for determining a piecewise linear function). Independently, Waterman *et al.* [20] developed a program that also finds the parametric decomposition. At the high level the methods are very similar, but important details and worst-case time bounds differ. Also, their work does not concern the questions addressed in this paper of the number and characteristics of the regions and lines in the decomposition.

We first consider a case of particular interest, the two-dimensional case where gaps are ignored (that is, $\gamma = 0$). Here, the optimal alignment is the one that maximizes $w - \alpha x - \beta y$. We show that when *global* alignments are considered, the decomposition of the α, β space is surprisingly structured. We prove that the number of (convex) polygons in the decomposition is bounded by $n^{2/3}$. We show further that every polygon is infinite and is bounded by two rays, each of which runs along a line of the form $\beta = c + \alpha(c + 0.5)$ for some constant c. As a consequence, we show that the *entire* decomposition can be simply found in $O(knm)$ time, where k is the number of actual polygons in the decomposition. Hence, the amortized cost for finding a single region is $O(nm)$ time, which is also the time to find a *single* optimal alignment within that region. For *local* alignments, the decomposition becomes more complex; some of its regions can be bounded (finite), but their total number never exceeds n^2. Any arbitrary line in the two-dimensional space can therefore go through at most n^2 regions. As a consequence, a simple implementation of the algorithm used in PARAL finds all the regions in time (n^3m) per region. A more involved, yet still practical, implementation of PARAL (which we have incorporated into a more user-friendly program called XPARAL) reduces that time to $O(nm)$ per region for all penalty functions except the one discussed in Section 5. An alternative method by Gusfield [8], which is an adaptation of Megiddo's method [11], can find all regions in time $O(nm \log^3 n)$ per region for any of the penalty functions. These algorithms are only briefly sketched in this paper and will be described in more details in a subsequent paper.

If gaps are allowed, then we obtain similar results for the decomposition of the three-dimensional space if *global* alignments are considered. All regions, which are convex polyhedra, are unbounded cones, bordered by rays of the form $\beta = c + (c + 0.5)\alpha$, $\gamma = d + d\alpha$. An arbitrary line in the three-dimensional space can go through at most n^2 regions if *global* alignments are computed, and through at most n^3 regions for *local* alignments.

A much more complex case arises when different weights are assigned to every possible mismatch (for example, in amino acid sequences, where the alphabet size is 20, there are 190 possible mismatch weights). In this most general setting, we show a subexponential bound on the number of regions that any straight line can intersect.

2. Two-Parameter Global Alignment. In this section we consider the global alignment problem where only two parameters α and β are given. We ignore the number of gaps, so $\gamma = 0$. The objective function, therefore, is to maximize $w - \alpha x - \beta y$, where w, x, and y are the numbers of matches, mismatches, and indels, respectively. We are interested in bounding the number of regions in the (first quadrant of) the α, β plane. We establish the following lemmas and observations for this purpose.

LEMMA 2.1. *For any alignment $\mathscr{A}$ with corresponding tuple (w, x, y), $2w + 2x + y = N$, where $N = n + m$ is the sum of the sequence lengths.*

PROOF. Any character can be part of exactly one match, mismatch, or indel. A match or a mismatch involves two characters. Thus the total number of characters that form part of a match is $2w$. Similarly, the total number of characters involved in mismatches is $2x$. An indel involves only one character from the input sequences and since we do not ignore any indels in counting their total number it follows that the number of characters involved in indels is y. The lemma follows. □

Recall that $m > n$.

LEMMA 2.2. *For any alignment $\mathscr{A}$, $w + x \leq n$.*

PROOF. A match or mismatch involves one character from *each* sequence. Hence their total cannot exceed the number of characters in the shorter sequence. □

COROLLARY 2.1. *For any alignment $\mathscr{A}$, $m - n \leq y \leq m + n$.*

COROLLARY 2.2. *In all alignments of two sequences, y is always odd or always even depending on whether $m + n$ is odd or even. There are therefore only $n + 1$ distinct values for y.*

THEOREM 2.1. *Any line forming a boundary between two regions is of the form $\beta = c + (c + \frac{1}{2})\alpha$, for some $c > -\frac{1}{2}$.*

PROOF. At a given point (α, β) an alignment has the value $v = w - \alpha x - \beta y$. The left-hand side of Lemma 2.1, which can be rewritten as $w + x + y/2 = (n + m)/2$, is also a linear combination of w, x, and y, with $\alpha = -1$ and $\beta = -\frac{1}{2}$. This suggests that this point is of some significance. Therefore consider the point $(-1, -\frac{1}{2})$ on the α, β plane. At this point all alignments have the same value: $v = w + x + y/2 = (n + m)/2$ (from Lemma 2.1). In other words, *all the value planes must meet at* $\alpha = -1, \beta = -\frac{1}{2}, v = (n + m)/2$. It follows then that any intersection between two such planes must also pass through that point. Now, let $\beta = c + c_1\alpha$ (for some c_1 and c) be a boundary (intersection) line. Since $(-1, -\frac{1}{2})$ is a point on that line, $-\frac{1}{2} = c - c_1$. Hence $c_1 = c + \frac{1}{2}$. In other words, $\beta = c + (c + \frac{1}{2})\alpha$, for some c. Since we are only interested in the quadrant where β and α are positive, it follows that any line passing through that quadrant and the point $(-1, -\frac{1}{2})$ must have a positive slope. Hence $c > -\frac{1}{2}$. □

COROLLARY 2.3. *All the regions of optimality are semi-infinite regions bounded by lines of the form $\beta = c + (c + \frac{1}{2})\alpha$ or by the coordinate axes.*

An example (using two made-up sequences) of the decomposition of the parameter space into regions of optimality, displaying the above property, is shown in Figure 1.

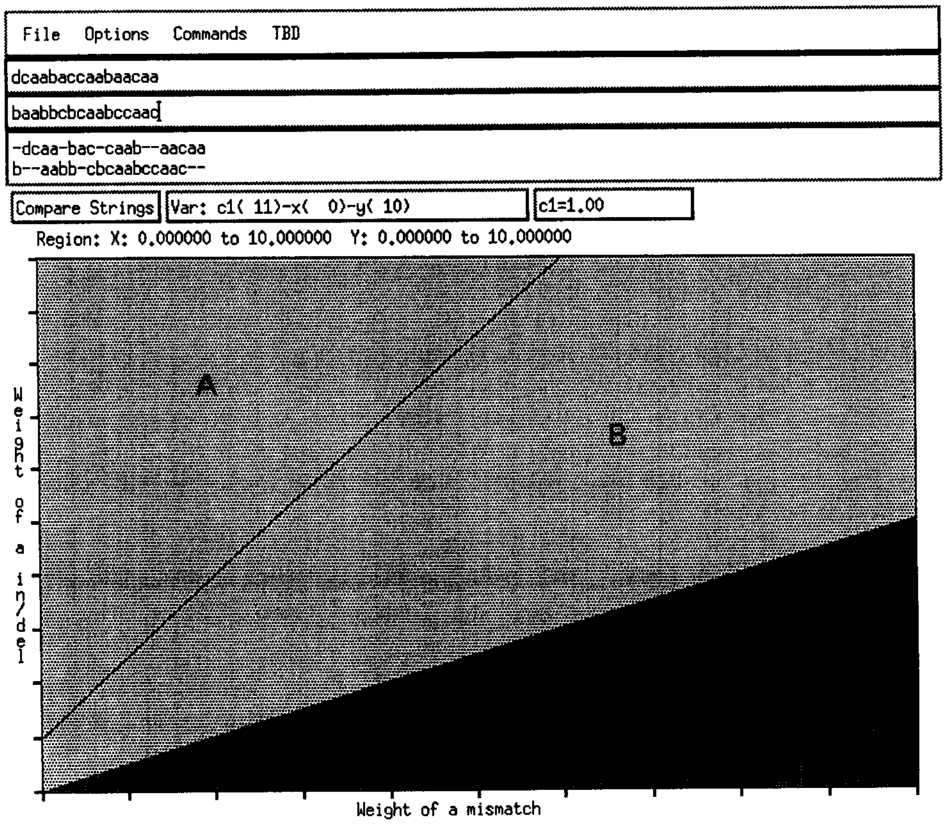
2.1. The Number of Regions. We now examine the maximum number of regions in any decomposition. A *breakpoint* along any given line is the point where the line moves between two adjacent regions.

LEMMA 2.3. *Along any horizontal line we never encounter breakpoints in the region $\alpha > 2\beta$.*

PROOF. Consider an alignment which contains at least one mismatch. A single mismatch may always be replaced by two indels, one in each sequence, so an alignment containing mismatches can be changed into one with no mismatches without affecting the number of matches. Thus if the cost of a mismatch, α, is greater than twice that of an indel, β, it follows that any optimal alignment for those parameters will have no mismatches.

Consider now a horizontal line in the region $\alpha > 2\beta$. Any alignment that is optimal at a point in this region can have no mismatches. Thus the value of such an optimal alignment remains constant through this region on any horizontal line (where β is constant). Hence there are no breakpoints on a horizontal line in this region. □

LEMMA 2.4. *There are at most $n + 1$ regions.*



```
dcaabaccaabaacaa
baabbcbcaabccaac
-dcaa-bac-caab--aacaa
b--aabb-cbcaabccaac--
Var: c1( 11)-x( 0)-y( 10)
c1=1.00
Region: X: 0.000000 to 10.000000  Y: 0.000000 to 10.000000
```

Fig. 1. Parametric decomposition of the (α, β) space for global alignments of the strings *dcaabaccaabaacaa* and *baabbcbcaabccaac*. The value functions for the various regions are: A, $v(\alpha, \beta) = 7 - 9\alpha - 0\beta$; B, $v(\alpha, \beta) = 11 - 3\alpha - 4\beta$; C, $v(\alpha, \beta) = 11 - 0\alpha - 10\beta$. The darkened region corresponds to the alignment shown above the decomposition.

Proof. Lemma 2.3, with $\beta = 0$, shows that we will encounter no breakpoints along the α axis and that therefore all the region boundaries must intersect the positive β axis. In other words, the β axis intersects all the regions. Let the alignments encountered, in order of increasing β, be $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_{k+1}$. Since $\alpha = 0$ the value of an alignment (along this line) is simply $v_i(\beta) = w_i - \beta y_i$. Since our objective function aims to maximize the alignments value, it can be seen that $y_{i+1} < y_i$ for all $\mathscr{A}_i$ $(i < k)$. By Corollary 2.2, the y_i can attain only $n + 1$ distinct values and the lemma follows. □

Let $w_i - \alpha x_i - \beta y_i$ and $w_j - \alpha x_j - \beta y_j$ be the planes corresponding to the values of two alignments $\mathscr{A}_i$ and $\mathscr{A}_j$, respectively. The equation of the line (or line segment) forming the boundary between the two regions with alignments $\mathscr{A}_i$ and $\mathscr{A}_j$ is

$$\beta = \frac{w_i - w_j}{y_i - y_j} + \frac{x_j - x_i}{y_i - y_j}\alpha.$$

We call this the *ratio form* of the boundary line.[3] The ratio form of the boundary line suggests an added constraint on the number of mismatches x, and this constraint can be used to refine Lemma 2.4 further.

THEOREM 2.2. *The number of regions is bounded by* $O(n^{2/3})$.

PROOF. We have seen that the boundaries between regions are of the form $\beta = c + (c + \frac{1}{2})\alpha$. Thus we may specify a boundary simply by specifying the slope $m = c + \frac{1}{2}$. Consider the β axis which, as we have seen, intersects all the regions. Let the alignments encountered, in order of increasing β, be $\mathscr{A}_1, \mathscr{A}_2, \ldots, \mathscr{A}_{k+1}$. Let them be separated by boundary lines with slopes $m_1, m_2, \ldots, m_k$, respectively. We have seen that $y_{i+1} < y_i$ for all $\mathscr{A}_i$ $(i < k)$. The slopes m_i are positive since we are interested only in the quadrant where α and β are positive, and any line from the point $(-1, -\frac{1}{2})$ that intersects that quadrant must have a nonzero (and noninfinite) positive slope. Let $\Delta x_i = x_{i+1} - x_i$ and $\Delta y_i = y_i - y_{i+1}$. Since $y_{i+1} < y_i$, Δy_i is positive. From the ratio form of the boundary line, $m = \Delta x_i / \Delta y_i$, therefore, Δx_i is positive. Thus a boundary slope can be identified by a pair of values $(\Delta x_i, \Delta y_i)$, and there can be no more boundaries than there are distinct $(\Delta x_i, \Delta y_i)$ pairs.

Now, $\sum_i \Delta x_i \le n$ and $\sum_i \Delta y_i \le (m + n) - (m - n) = 2n$. Therefore

$$3n \ge \sum_{i=1}^{k} (\Delta x_i + \Delta y_i).$$

For all t consider the number of boundaries such that $(\Delta x_i + \Delta y_i) = t$. Since the pairs have to be distinct there can be only $t - 1$ such pairs. Intuitively, the way to maximize the number of $(\Delta x_i, \Delta y_i)$ pairs is to generate as many as possible which sum to two, then three, and so on as long as the constraint on the total is maintained. Let s be the largest value such that

$$3n \ge \sum_{t=2}^{s} t(t-1)$$
$$= \tfrac{1}{3}(s-1)s(s+1).$$

Therefore $s = O(n^{1/3})$. The number of regions is therefore

$$k \le \sum_{t=1}^{s+1} (t-1) = \frac{s(s+1)}{2} = O(s^2) = O(n^{2/3}). \qquad \square$$

[3] Writing the intersection of the two planes in this way also gives an interesting interpretation of the boundary between two regions. The slope of the boundary is the rate at which mismatches are exchanged for indels in the two adjacent alignments, and the intercept of the boundary line is the rate at which matches are exchanged for indels.

A closer analysis based on the same observation and using Euler's function $\Phi(i)$, the number of integers less than and relatively prime to i, can be used to show that $k \leq 0.88n^{2/3}$.

THEOREM 2.3. *All regions in the decomposition can be simply found in $O(nm)$ time per region, i.e., the time per region is no more than the time to compute a single alignment at a fixed α, β point.*

PROOF. The Eisner–Severance [1], or Newton's, method finds all the breakpoints (intersections between regions) along any single line or direction in $O(nm)$ time per breakpoint. We have seen that a single line, the β axis, intersects all the regions. Hence all the regions can be found in $O(knm)$ time by the Eisner–Severance method where k is the number of regions. This actually gives us the intercepts along the β axis but this information is enough to determine the boundary lines since each line must pass through $(-1, -\frac{1}{2})$. $\square$

COROLLARY 2.4. *The entire decomposition can be found in $O(n^{5/3}m)$ time.*

3. Two-Parameter Local Alignment.

3. Two-Parameter Local Alignment. In this section we consider the regions of optimality generated by looking at local instead of global alignments. In this case certain spaces, mismatches, and matches occurring at the ends of the alignment may be disregarded. This renders Lemma 2.1 invalid. However, we note that Lemma 2.2 remains valid. We may also make the following weaker observation.

LEMMA 3.1. *For any local alignment,*

$$2w + 2x + y \leq N,$$

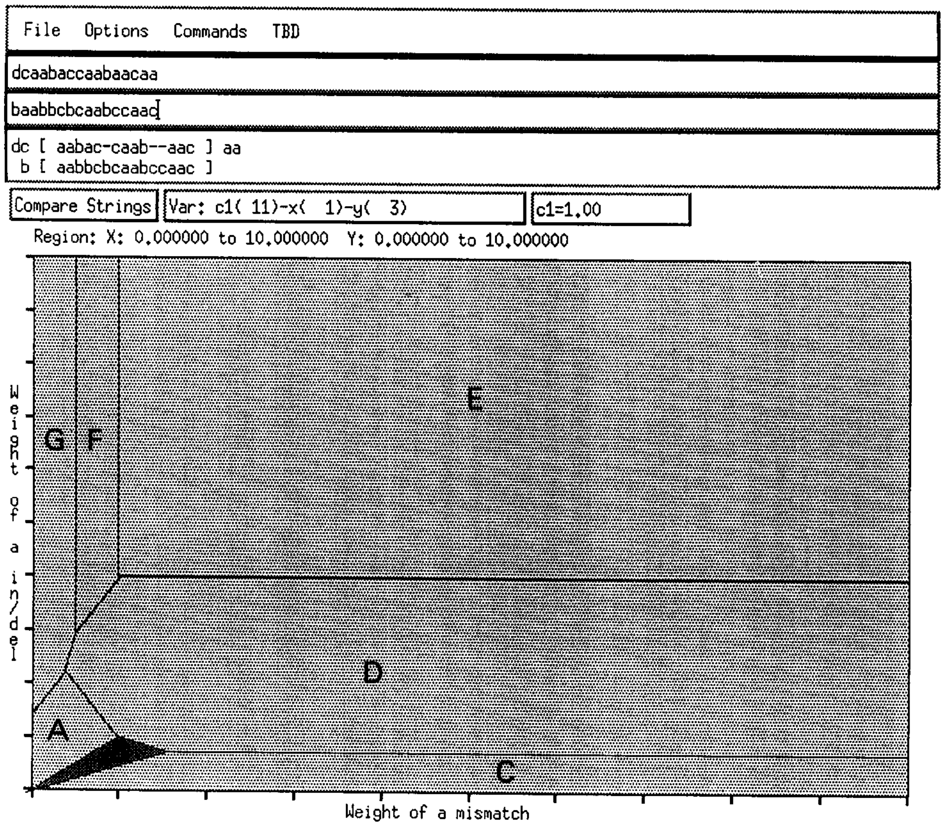
where $N = n + m$ is the sum of the sequence lengths.

In the local (or end-free) case, "extreme" spaces *will not* be counted (since any additional space will always decrement the value). Thus if space is "counted," it must be the case that there is at least one match/mismatch on either side of it, so if $y > 0$, then we must have at least two matches/mismatches consuming four characters. Hence,

LEMMA 3.2. *For any alignment,*

$$y \leq m + n - 4.$$

Due to these weaker conditions, the picture of all the optimal regions in the local alignment case may be more complicated and not show the structure that we observed for global alignments. The example in Figure 2, using the same sequences as Figure 1, illustrates this increased complexity.



```
dcaabaccaabaacaa
baabbcbcaabccaac

dc [ aabac-caab--aac ] aa
 b [ aabbcbcaabccaac ]

Var: c1( 11)-x( 1)-y( 3)
c1=1.00
Region: X: 0.000000 to 10.000000 Y: 0.000000 to 10.000000
```

Fig. 2. Parametric decomposition of the (α, β) space for local alignments of the strings *dcaabaccaabaacaa* and *baabbcbcaabccaac*. The value functions for the various regions are: A, $v(\alpha, \beta) = 11 - 2\alpha - 2\beta$; B, $v(\alpha, \beta) = 11 - 1\alpha - 3\beta$; C, $v(\alpha, \beta) = 11 - 0\alpha - 5\beta$; D, $v(\alpha, \beta) = 8 - 0\alpha - 1\beta$; E, $v(\alpha, \beta) = 4 - 0\alpha - 0\beta$; F, $v(\alpha, \beta) = 6 - 2\alpha - 0\beta$; G, $v(\alpha, \beta) = 8 - 6\alpha - 0\beta$. Region B corresponds to the local alignment given above the decomposition.

We further note that Lemma 2.3 remains valid in the local alignment case since it did not require any of the above-mentioned conditions for its proof.

LEMMA 3.3. *There are at most n^2 optimal regions.*

PROOF. Consider two alignments $\mathcal{A}_1$ and $\mathcal{A}_2$, each optimal for some region in the decomposition, which have tuples (w, x, y_1) and (w, x, y_2), i.e., they differ only in the number of indels. Without loss of generality, assume that $y_1 < y_2$. Since we are interested in the region where the mismatch and indel penalties are always positive, it follows that $\mathcal{A}_1$ will always have a larger value than $\mathcal{A}_2$, which contradicts our assumption that $\mathcal{A}_2$ is optimal for some region.

In other words, it is not possible for two different alignments to have the same values of w and x. Thus there can be only as many regions as there are distinct pairs w, x. This proves the lemma. □

Two Parameters with Gaps. Suppose that we allow gaps in the objective function, but still hold the number of variable parameters to two. An example of this is the important objective function: Maximize

$$v = c_1 w - c_2 x - \alpha y - \beta z,$$

where c_1 and c_2 are constants. Then the number of optimal regions in the decomposition is at most $O(nm)$, for essentially the same reasons. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two alignments that are optimal for two regions in the decomposition with tuples (w_1, x_1, y, z) and (w_2, x_2, y, z). If, say, $c_1 w_1 - c_2 x_1 > c_1 w_2 - c_2 x_2$, then $\mathcal{A}_1$ will always have a higher value than $\mathcal{A}_2$, a contradiction. Hence, it is not possible for two different alignments in the decomposition to have the same values of y and z, so the number of regions is bounded by the maximum number of distinct (y, z) pairs, which is $O(nm)$ for the local case (in the next section we show that $z \leq 2n - 1$). This fact is important in obtaining a time bound of $O(nm)$ per polygon for the algorithm that finds the entire decomposition.

4. Three-Parameter Alignment.

We now consider the case where gap penalties can also vary as a parameter γ, hence the parameter space is now three-dimensional. The consideration of gap penalties is very important in the context of DNA or protein sequences since, in many cases, the alignments accepted as "standard" or best by biologists cannot be obtained by the dynamic programming approach unless a specific nonzero penalty is added for each gap. For an example of this see [5]. As in the two-dimensional case, we can prove a simpler decomposition when global alignments are considered. Recall that z is the number of gaps in the alignment. The following observation holds in general:

LEMMA 4.1. $z \leq 2n - 1$.

PROOF. Any space that is introduced in the long sequence must be opposite to a character in the short sequence, hence the number of spaces, and therefore the number of gaps, in the long sequence is bounded by n. Likewise, any gap in the short string must be bracketed by characters at both ends, hence the number of gaps in the short string is bounded by $n - 1$. The claim follows. □

4.1. Global Alignments with Gaps. We first note that Lemmas 2.1 and 2.2 and Corollaries 2.2 and 2.1 from Section 2 are still valid in this three-parameter global-alignment case, for exactly the same reasons, since we are not changing the way we count indels, mismatches, or spaces.

Let us describe a line in three dimensions α, β, and γ as a function of one parameter, α, by two equations $\beta = c_0 + c_1\alpha$ and $\gamma = c_2 + c_3\alpha$.

THEOREM 4.1. *Any line forming a boundary between three or more regions is of the form* $\beta = c + (c + \frac{1}{2})\alpha$, $\gamma = d + d\alpha$.

PROOF. Consider the point $(-1, -\frac{1}{2}, 0)$ on the α, β, γ plane. At this point all the alignments have the same value: $v = w + x + y/2 = (n + m)/2$ (from Lemma 2.1). In other words, all the value hyperplanes must meet at $\alpha = -1$, $\beta = -\frac{1}{2}$, $\gamma = 0$, $v = (n + m)/2$. By the same reasoning as in Theorem 2.1 it follows that the intersection between two such hyperplanes is a plane containing this point and the intersection between three or more hyperplanes must be a line passing through this point. Again using the same reasoning it can be seen that the family of these boundary lines passing through the point $\alpha = -1$, $\beta = -\frac{1}{2}$, $\gamma = 0$ is described by the conditions $\beta = c + (c + \frac{1}{2})\alpha$, $\gamma = d + d\alpha$. □

COROLLARY 4.1. *All optimal regions are semi-infinite "conic" regions bounded by lines of the form* $\beta = c + (c + \frac{1}{2})\alpha$, $\gamma = d + d\alpha$.

This implies that just as the two-parameter global alignment essentially had only one degree of freedom (in the sense that alignments that are optimal at some point must also be optimal at some point on the α or β axis) so the three-parameter global alignment also essentially has only two degrees of freedom since any alignment that is optimal at some interior point must also be optimal on one of the three coordinate planes where one of the three parameters is zero. In other words, if we compute the decomposition for the three coordinate planes we have all the information required to describe the entire decomposition.

THEOREM 4.2. *There are at most* $O(n^2)$ *optimal regions in the decomposition of the* α, β, γ *parameter space.*

PROOF. We know from Theorem 2.2 that there can be only $O(n^{2/3})$ regions on the $\gamma = 0$ plane. Using the same arguments as in Lemma 3.3, we can see that on the $\beta = 0$ plane there can be at most as many regions as there are distinct (w, x) pairs and on the $\alpha = 0$ plane there can be at most as many regions as there are distinct (w, y) pairs. Both of these are n^2. □

By similar arguments, it is easy to show that, in the case of *local alignments with gaps*:

(1) An arbitrary line can go through at most n^3 regions.
(2) The number of regions is at most n^3.

5. The Case of Richer Weights and Penalties.

In the previous results the total penalty for mismatches was just the product of the mismatch penalty α and the *number* of mismatches. While this is sufficient in many biological applications, many other applications use a richer set of weights and penalties. In detail, for each pair (a, b) of unequal characters in the alphabet, there is a number $w(a, b)$ which is the base penalty for aligning these mismatching characters. A character-dependent penalty for aligning a particular character with a space, and also a positive weight for aligning two matching characters which depends on the particular pair of characters may also be specified. There are several commonly used pair-dependent weight and penalty schemes in the biological literature. The most widely referred to is the PAM matrix, developed by Dayhoff [15].

With such pair-dependent weights and penalties, the value of an alignment $\mathscr{A}$ is computed as $M(\mathscr{A}) - MS(\mathscr{A}) - S(\mathscr{A})$ where $M(\mathscr{A})$ is the sum of all the (pair-dependent) weights contributed by matching pairs of characters in $\mathscr{A}$, $MS(\mathscr{A})$ is the sum of all the (pair-dependent) penalties contributed by mismatching pairs of characters, and $S(\mathscr{A})$ is the sum of all the (pair-dependent) penalties contributed by characters opposite spaces.

One might want to parametrically study the effect of changing these pair-dependent weights, but this seems too unwieldy. A simpler question that is still of importance is how to balance the influence of the term contributed by matches versus the terms contributed by mismatches and spaces. So for a given alignment $\mathscr{A}$, its parametric value is $M(\mathscr{A}) - \alpha MS(\mathscr{A}) - \beta S(\mathscr{A})$. As before, the α, β space decomposes into maximal convex regions where a particular alignment is optimal throughout. The results in the previous sections depend on $M(\mathscr{A})$, $MS(\mathscr{A})$, and $S(\mathscr{A})$ being the *number* of matches, mismatches, and spaces in $\mathscr{A}$, respectively, but they do not carry over for a richer weight/penalty structure. We do not know nontrivial bounds on the number of regions in the parametric decomposition, but we can prove that along any line, the number is subexponential.

THEOREM 5.1. *With pair-dependent weights and penalties, the number of breakpoints encountered along any line L in the parametric decomposition is at most $(2m)^{\log_2 n}$.*

PROOF. Along a line L in α, β space, the value of β is linearly dependent on α, so by adjusting the base penalties for spaces, we have a one-parameter problem. In that problem the parametric value of an alignment $\mathscr{A}$ can be assumed to be $M(\mathscr{A}) - \alpha[MS(\mathscr{A}) + S(\mathscr{A})]$. Consider the dynamic programming table used to find the optimal alignment, once a fixed value of α is specified. An optimal alignment is specified by a path in that table from cell $(0, 0)$ to cell (n, m). We associate a single optimal alignment in each region of the decomposition, and hence a single optimal path in each region. Thus as we move along L (with changing α) through changing regions, the corresponding path changes. Let S be the set of paths which correspond to the regions encountered along L. Let $T(n, m)$ denote the maximum possible size of S in any n by m table.

Each path in S goes through row $n/2$. Consider a fixed cell $(n/2, k)$. The number of paths in S which go through cell $(n/2, k)$ is bounded by

$$T(n/2, k) + T(n/2, m - k) \leq 2T(n/2, m).$$

The reason for the plus (rather than a product) is that changes in the optimal path before row $n/2$ occur as α is changing and are totally independent with changes in the optimal path that occur after row $n/2$. Hence $T(n, m) \leq 2mT(n/2, m)$, which implies that $T(n, m) \leq (2m)^{\log_2 n}$. □

6. Program Description. We have developed a program, PARAL, which allows the user to specify two sequences, a range for α and β, and the desired type of alignment, global, local, or end-free, with or without gaps. Then, when a specific choice for α and β is given, the program computes an optimal alignment $\mathscr{A}$ for that choice and then determines and displays the region P in the α, β space for which $\mathscr{A}$ is optimal. The user may explore the interesting part of the space by repeatedly specifying values for α and β which have not been placed yet in a region. It can also generate all the regions in the entire decomposition systematically without having to choose any specific points. A version of PARAL, called XPARAL, now runs under X-windows. A program for parametric sequence comparisons is also reported in [20].

The implementation is based on the following primitive: given a point p and a direction l, find the first point p' along l in which l crosses to a different region, and also find the new alignment at p'. This primitive can be simply implemented in $O(knm)$ time, where k is the actual number of regions that l goes through, by using the method of [4] and [17] that finds all breakpoints along a line. It can also be implemented in $O(nm \log^3 n)$ time, independent of k, where each successive breakpoint is found by Gusfield's [8] adaptation of Megiddo's method [11]. We have adopted the first approach in PARAL.

Given a point p, its optimal region P is ideally found as follows: First, an arbitrary direction l is chosen and the next point p' is computed. Given the alignment at p', one boundary of P can now be determined by intersecting both alignments. The procedure is repeated, say, clockwise, along the new boundary until all boundaries of P are found. The idealized method needs more detail to handle degeneracies that can occur if more than three regions meet at a point. We omit the details here.

By more careful bookkeeping, we can implement the algorithm so that every alignment computed gets "charged" to either a polygon, a line segment, or a vertex of the decomposition, with the guarantee that at most a constant number of alignments are charged to any polygon, line segment, or vertex. The idea is that we keep information about every alignment done and use that list to avoid redundant alignments, hence speeding up the basic algorithm of [4] and [17]. The net result is that the (amortized) running time is $O(R + nm)$ per polygon, where R is the total number of polygons in the decomposition. Since R is at most n^2 or nm in all the two-parameter objective functions, the running time is $O(nm)$ per polygon. The same bound of $O(R + nm)$ per polygon holds in the case of the richer objective functions considered in Section 5 but this does not imply the $O(nm)$-time bound per polygon since R in that case is not known to be bounded by $O(nm)$.

A more complete paper on the program is in progress [9].

Acknowledgments. Thanks to Rob Irving for working out the constant in the $O(n^{2/3})$ bound on the number of regions in the case of two-dimensional global alignment (Theorem 2.2). Thanks also to Stephen Altschul who encouraged us to re-examine the implementation of PARAL, leading to the amortized improvement mentioned in Section 6.

References

[1] P. Argos and M. Vingron, Sensitivity comparison of protein amino acid sequences, in *Methods in Enzymology*, Vol. 183 (R. Doolittle, ed.), Academic Press, San Diego, CA, pp. 352–365.

[2] R. F. Doolittle, *Of Urfs and Orfs: A Primer on How To Analyze Derived Amino Acid Sequences*, University Science Books, 1986.

[3] R. F. Doolittle (ed.), *Methods in Enzymology*, Vol. 183, Academic Press, San Diego, CA.

[4] M. Eisner and D. Severance, Mathematical techniques for efficient record segmentation in large shared databases. *J. Assoc. Comput. Mach.*, **23** (1976), 619–635.

[5] W. M. Fitch and T. F. Smith, Optimal sequence alignments. *Proc. Nat. Acad. Sci. USA*, **80** (1983), 1382–1386.

[6] O. Gotoh, Optimal sequence alignment allowing for long gaps, *Bull. Math. Biol.*, **52**(3) (1990), 359–373.

[7] M. Gribskov and J. Devereux, *Sequence Analysis Primer*, Stokton Press, 1991.

[8] D. Gusfield, Parametric combinatorial computing and a problem of program module distribution, *J. Assoc. Comput. Mech.*, **30** (1983), 551–563.

[9] D. Gusfield, K. Balasubramanian, J. Bronder, D. Mayfield, D. Naor, and P. Stelling, PARAL: An efficient program to optimally align strings with variable match, mismatch, space and gap weights, in preparation.

[10] D. Gusfield, K. Balasubramanian and D. Naor, Parametric optimization of sequence alignment, *Proceedings of the Third Annual ACM–SIAM Symposium on Discrete Algorithms*, 1992, pp. 432–439.

[11] N. Megiddo, Combinatorial optimization with rational objective functions, *Math. Oper. Res.*, **4** (1979), 414–424.

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

[13] D. Sankoff and J. Kruskal (eds.), *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Addison-Wesley, Reading, MA, 1983.

[14] G. D. Schuler, S. F. Altschul and D. J. Lipman, A workbench for multiple alignment construction and analysis, in *Proteins: Structure Function and Genetics*, **9**(3), 180–190, in press.

[15] R. Schwarz and M. Dayhoff, Matrices for detecting distant relationships, in *Atlas of Protein Sequences*, National Biomedical Research Foundation, Washington, DC, 1979, pp. 353–358.

[16] T. F. Smith and M. S. Waterman, Identification of common molecular subsequences, *J. Molecular Biol.*, **147** (1981), 195–197.

[17] H. Stone, Critical load factors in distributed systems. *IEEE Trans. Software Engrg.*, **4**(3) (1978), 254–258.

[18] G. von Heijne, *Sequence Analysis in Molecular Biology*, Academic Press, New York, 1987.

[19] M. S. Waterman, Sequence alignments, in *Mathematical Methods for DNA Sequences* (M. S. Waterman, ed.), CRC Press, Boca Raton, FL, 1989, pp. 53–92.

[20] M. S. Waterman, M. Eggert, and E. Lander, Parametric sequence comparisons, *Proc. Nat. Acad. Sci. USA*, **89** (1992), 6090–6093.