CABIOS

Vol.8. no.5. 1992
Pages 511 – 520

Dynamic programming algorithms for restriction map comparison

Xiaoqiu Huang and Michael S.Waterman[1]

Abstract

For most sequence comparison problems there is a corresponding map comparison algorithm. While map data may appear to be incompatible with dynamic programming, we show in this paper that the rigor and efficiency of dynamic programming algorithms carry over to the map comparison algorithms. We present algorithms for restriction map comparison that deal with two types of map errors: (i) closely spaced sites for different enzymes can be ordered incorrectly, and (ii) closely spaced sites for the same enzyme can be mapped as a single site. The new algorithms are a natural extension of a previous map comparison model. Dynamic programming algorithms for computing optimal global and local alignments under the new model are described. The new algorithms take about the same order of time as previous map comparison algorithms. Programs implementing some of the new algorithms are used to find similar regions within the Escherichia coli restriction map of Kohara et al.

Introduction

It is often the case that a physical map is constructed prior to genomic sequencing projects, as in the pioneering work of Coulson *et al.* (1986) on nematode and of Olson *et al.* (1986) on yeast. Kohara *et al.* (1987) published a physical map of the entire *Escherichia coli* chromosome showing the approximate locations of the restriction sites for eight restriction enzymes. Many projects are underway to physically map a variety of genomes. Clearly physical maps will often be complete long before the complete sequence information is available. For a variety of reasons it is important to have appropriate analytical tools to study these data. In Waterman *et al.* (1984) the model problem was to study the evolution between homologous genes from closely related organisms. The goal is to align all of each map—the so-called global alignment problem. Later Rudd *et al.* (1990) posed the problem of locating sequenced DNA from *E.coli* on the Kohara map. Here a small and highly precise map is to be aligned somewhere along a very long map. Obviously many other variants exist; essentially there is a map alignment problem of interest for every sequence alignment problem. Finding the best local map alignments, for example, corresponds to the best local sequence alignment problem of Smith and Waterman (1981). Finding the best overlap between two mapped clones is relevant to construction of the genomic physical map itself and finding the best overlap between two sequence fragments is relevant to construction of the sequence itself. One of the major points of this paper is that the rigor and efficiency of dynamic programming algorithms can be retained even when the data appear incompatible with dynamic programming.

Restriction maps give a sequence of restriction sites with distances between the sites. If intersite distances were ignored, then dynamic programming algorithms for sequence comparison could be immediately applied to compare two restriction maps. Waterman *et al.* (1984) developed an alogrithm that handles the distances between sites as well as the linear sequence of sites. It is necessary, however, to examine restriction maps more carefully. Restriction map data often contain two types of errors: (i) the order of closely spaced sites for different enzymes is incorrect, and (ii) closely spaced sites for the same enzyme are merged as a single site. Map comparison methods that properly address the two types of map errors are likely to discover map similarities that are missed by methods that only deal with the insertions/deletions (indels) of sites and distance discrepancies.

Waterman *et al.* (1984) first defined a notion of map alignment and gave an algorithm for computing an optimal alignment of two maps. The alignment allowd for site indels as well as differences in distance between sites. This notion of alignment and scoring method do not handle the two types of map errors listed above. A simple modification to the algorithm handles the second type of errors, but not the first type (Waterman and Raymond, 1987). Myers and Huang (1992) improved the time efficiency of the algorithm of Waterman *et al.* (1984). Miller *et al.* (1990) introduced a notion of alignment that treats both types of map errors and designed a simple algorithm for computing an optimal alignment. A recent improvement of the Miller *et al.* algorithm permits the search of a long map of m sites for best matches to a short map of n sites to be done in $O(mn \log n)$ time (Miller *et al.*, 1991), which is about the time requirement of a faster version of a variant of the algorithm presented in this paper. The alogrithms of Miller *et al.* (1990, 1991) are used to align a sequence to a restriction map (Rudd *et al.*, 1990, 1991). However, the scoring scheme employed by Miller *et al.* (1990, 1991) can

Department of Computer Science, Michigan Technological University, Houghton MI 49931–1295 and [1]Departments of Mathematics and Molecular Biology, University of Southern California, Los Angeles, CA 90089–1113, USA

cause a single map error to be penalized many times. Also it is not clear how to generalize the algorithm to find best local map alignments efficiently. The notion of alignment and objective scoring function described in this paper address both types of errors and allow nearly all varients of map comparison problems to be solved efficiently. This is done through the generalization of the dynamic programming model of Waterman *et al.* (1984) by introducing the concept of a 'segment', which is a short region of the map where both types of errors may occur.

Next we present the map comparison model and the algorithm of Waterman *et al.* (1984). A map of m restriction sites is represented by a sequence of enzyme-position pairs $<a_1,q_1>$ $<a_2,q_2>$ . . . $<a_m,q_m>$. Here a_i denotes the enzyme recognizing site i and q_i gives the position of site i in base pairs relative to some origin. Assume that the pairs are ordered according to their positions, i.e. $q_i \leq q_{i+1}$, $1 \leq i < m$. Let $A = <a_1,q_1><a_2,q_2>$ . . . $<a_m,q_m>$ and $B = <b_1,r_1>$ $<b_2,r_2>$ . . . $<b_n,r_n>$ be two maps of m and n sites respectively. $A[i]$ denotes site i of A, and $A[s,t]$ denotes sites s through t of A, $s \leq t$.

An alignment Π of A and B is a sequence of ordered matching pairs of sites (i_1,j_1) (i_2,j_2) . . . (i_d,j_d), where $a_{i_t} = b_{j_t}$, $i_t < i_{t+1}$ and $j_t < j_{t+1}$ for each t. Let ν, λ and μ be non-negative numbers. Then the score of the global alignment Π is defined to be

$$\text{score}(\Pi) = \nu \times d - \mu \times |q_{i_1} - r_{j_1}| - \mu \times \sum_{t=2}^{d} |(q_{i_t} - q_{i_{t-1}}) - (r_{j_t} - r_{j_{t-1}})| - \mu \times |(q_m - q_{i_d}) - (r_n - r_{j_d})| - \lambda \times (m + n - 2d)$$

In this scoring scheme, each matching pair is given a reward of ν. The discrepancy in distance between adjacent aligned pairs (i_{t-1},j_{t-1}) and (i_t,j_t) is penalized by a factor of μ, as are the discrepancies to the leftmost and rightmost aligned pairs. Also each unaligned site is penalized by λ.

Assume throughout this paper that the maximum over the empty set is $-\infty$, i.e. $\max \varnothing = -\infty$. In practice, a proper substitute for $-\infty$ can be $-\mu(q_m + r_n) - \lambda(m + n)$, the score of the alignment of no matching pairs. For a pair (i,j) with $a_i = b_j$, let $X(i,j)$ be the score of the largest scoring alignment with the rightmost aligned pair (i,j), not including the term $\mu|(q_m - q_i) - (r_n - r_j)|$. Then we have the following recurrences for computing the matrix X.

$$Y(i,j) = \max\{X(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| : g < i, h < j \text{ and } a_g = b_h\}$$

$$X(i,j) = \max\{\nu - \lambda(m + n - 2) - \mu|q_i - r_j|, Y(i,j) + \nu + 2\lambda\}$$

Here $\nu - \lambda(m + n - 2) - \mu|q_i - r_j|$ is the score of the alignment consisting only of the pair (i,j), and $Y(i,j) + \nu + 2\lambda$ is the score of the largest scoring alignment with at least one matching pair to the left of (i,j). Both $X(i,j)$ and $Y(i,j)$ exclude the end penalty $\mu|(q_m - q_i) - (r_n - r_j)|$. The score of an optimal alignment is therefore

$$\max\{X(i,j) - \mu|(q_m - q_i) - (r_n - r_j)| : 1 \leq i \leq m, 1 \leq j \leq n \text{ and } a_i = b_j\}$$

A simple dynamic programming algorithm as shown in Figure 1 computes the maximum score in $O(m^2n^2)$ time.

A simple way to speed up the computation is to limit the number of X-values examined in computing $Y(i,j)$ to at most δ^2 for some integer δ. The two nested **for** loops indexed by g and h in Figure 1 become

for $g \leftarrow \max\{1, i - \delta\}$ **to** $i - 1$ **do**
  for $h \leftarrow \max\{1, j - \delta\}$ **to** $j - 1$ **do**

This gives an $O(\delta^2 mn)$ approximate algorithm. Myers and Huang (1992) showed that the maximum alignment score of two maps can be computed in $O(mn(\log m + \log n))$ time.

If we change the definition of alignment so that a site of one map is allowed to match several closely spaced sites of the other map for the same enzyme, then a simple modification to the algorithm of Waterman *et al.* (1984) can compute an optimal alignment under this new model. Specifically, let the score of aligning $A[i]$ with $B[t,j]$ for the same enzyme be $\nu - \mu(r_j - r_t) = \nu - \mu[(r_{t+1} - r_t) + \ldots + (r_j - r_{j-1})]$. In Figure 1, insert the following statement immediately after the assignment statement with $X(i,j)$ on the left-hand side

$$X(i,j) \leftarrow \max\{X(i,j), \lambda + \max\{X(i,j-1) - \mu(r_j - r_{j-1}), X(i-1,j) - \mu(q_i - q_{i-1})\}\}$$

where $X(s,t) = -\infty$ if $a_s \neq b_t$. In the expression on the right-hand side, the term $\lambda + X(i,j-1) - \mu(r_j - r_{j-1})$ is the score of an optimal alignment with $A[i]$ aligned with $B[t,j]$ for some $t < j$. During the computation, no special care is needed to avoid meaningless alignments such as one where $A[i-1]$ is aligned with $B[j-1,j]$, and $B[j]$ with $A[i-1,i]$. The reason is that this alignment cannot be optimal. This modification first appeared in Waterman and Raymond (1987).

```
score ← −μ(q_m + r_n) − λ(m + n)
for i ← 1 to m do
    for j ← 1 to n do
        if a_i = b_j then
          { y ← −μ(q_m + r_n) − λ(m + n)
            for g ← 1 to i−1 do
                for h ← 1 to j−1 do
                    if a_g = b_h then
                        y ← max{y, X(g,h) − μ|(q_i − q_g) − (r_j − r_h)|}
            X(i,j) ← max{ν − λ(m + n − 2) − μ|q_i − r_j|, y + ν + 2λ}
            score ← max{score, X(i,j) − μ|(q_m − q_i) − (r_n − r_j)|}
          }
```

Fig. 1. The algorithm of Waterman *et al.* (1984).

System and methods

The programs are written in C and developed on Sun workstations. The programs are portable and are designed to run on workstations.

Algorithm

The extension of the basic model along with corresponding algorithms for global and local similarities is discussed. Then time-efficient algorithms for global and local alignments are described.

Global similarities

We introduce a new notion of alignment of two entire maps and describe a simple dynamic programming algorithm for computing the maximum score. Then we consider a space-efficient way to construct an alignment with the maximum score.

A dynamic programming algorithm. For some non-negative number α, we assume that if two sites for different enzymes are more than α base pairs apart, then the relative order of the two sites is always determined correctly. Also assume that two sites for the same enzyme that are more than α base pairs apart are always mapped as two distinct sites. In other words, two closely spaced sites for different enzymes may be ordered incorrectly, and two closely spaced sites for the same enzyme may be mapped as a single site. If two maps A and B are constructed using different methods, it might be necessary to use two distinct numbers α_1 for A and α_2 for B. For convenience, we develop the algorithms with $\alpha_1 = \alpha_2$.

A segment $[i,k]$ of A consists of sites i through k with $0 \leq q_k - q_i \leq \alpha$. A matching pair of segments $[i,k]$ of A and $[jl]$ of B is denoted by $(i,j;k,l)$. A global alignment Π of A and B is a sequence of ordered matching pairs of segments $(i_1,j_1;k_1,l_1)$ $(i_2,j_2;k_2,l_2)$ . . . $(i_d,j_d;k_d,l_d)$, where $k_t < i_{t+1}$ and $l_t < j_{t+1}$ for each $t < d$. An example of alignment illustrated in Figure 2 is represented by (1,2;3,4) (4,5;4,5) (5,6;7,8), where $\alpha = 30$. The similarity score $\sigma(i,j;k,l)$ of a matching segment pair $(i,j;k,l)$ is defined below. Let λ and μ be non-negative numbers. Then the score of the alignment Π is defined to be

$$\text{score}(\Pi) = \sum_{t=1}^{d} \sigma(i_t,j_t;k_t,l_t) - \mu|q_{i_1} - r_{j_1}| - \mu \sum_{t=2}^{d}$$
$$|(q_{i_t} - q_{k_{t-1}}) - (r_{j_t} - r_{l_{t-1}})| - \mu|(q_m - q_{k_d}) - (r_n - r_{l_d})| -$$
$$\lambda[m + n - \sum_{t=1}^{d} (k_t - i_t + 1) - \sum_{t=1}^{d} (l_t - j_t + 1)]$$

Here the discrepancy in distance between two adjacent aligned pairs $(i_{t-1},j_{t-1}; k_{t-1},l_{t-1})$ and $(i_t,j_t; k_t,l_t)$ is penalized by a factor of μ, as are the discrepancies to the leftmost and rightmost aligned pairs. Also each site not present in any aligned segment is penalized by λ.

For a finite set Q, let $||Q||$ denote the number of elements in Q. Let ν be a non-negative number. The score $\sigma(i,j;k,l)$ is defined to be

$$\sigma(i,j;k,l) = \nu||\{a_t : i \leq t \leq k\} \cap \{b_t : j \leq t \leq l\}|| -$$
$$\mu|(q_k - q_i) - (r_l - r_j)| - \lambda||\{t : i \leq t \leq k \text{ and } a_t \neq b_s$$
$$\text{for each } s,\ j \leq s \leq l\}|| - \lambda||\{t : j \leq t \leq l \text{ and } b_t \neq a_s$$
$$\text{for each } s,\ i \leq s \leq k\}||$$

Each enzyme common to both the segments $[i,k]$ and $[j,l]$ is given a reward of ν, and each site for an enzyme in one but not both of the segments is penalized by λ. Also the discrepancy in distance between the segments is penalized by a factor of μ. For example, in Figure 2, the score $\sigma(1,2;3,4)$ is $2\nu - \mu - \lambda$, and the score of this alignment is $6\nu - 4\lambda - 38\mu$. This definition allows sites for a common enzyme in one segment to be aligned with sites for the same enzyme in the other segment. Note that no order restriction is imposed on aligned sites between a pair of segments. This implies that a site in one segment may be aligned with several sites in the other segment and that aligned pairs of sites need not have the same order in each segment. In other words, this definition accommodates the two types of map errors.

The definition of the function σ given in the previous paragraph is intended for the situation where both maps contain the errors. Sometimes one of the two maps is error-free; for example, a map is constructed from a DNA sequence that is presumably correct. In such a situation, the definition of the scoring function can be adjusted slightly to reflect this. Assume that the map A is error-free and that the map B may contain the errors. We must not allow any site of A to be aligned with several sites of B for the same enzyme since no single site of A represents multiple sites. On the other hand, a single site of B may be aligned with several closely spaced sites of A for the same enzyme, and two nearby sites of B (for different enzymes) may be out of order with two nearby sites of A. For each common enzyme e between segments $[i,k]$ and $[j,l]$, if the number $c(e,i,k)$ of sites for the enzyme e in $[i,k]$ is larger than the number $f(e,j,1)$ of sites for e in $[j,l]$, then $c(e,i,k) - f(e,j,l)$ sites for the enzyme e in the segment $[i,k]$ are deleted at a cost of λ each. Otherwise, there is no charge for these sites for e in $[i,k]$. Let $\gamma(x,y)$ be $x - y$ if $x > y$ and 0 otherwise. Then the scoring function σ' is defined as

$$\sigma'(i,j;k,l) = \sigma(i,j;k,l) - \lambda \sum_{\text{common enzyme } e} \gamma(c(e,i,k),f(e,j,l))$$

The function σ' gives the score of aligning each pair of segments from the maps A and B, where A is error-free but B is subject to the errors as discussed above.

An optimal global alignment has the maximum score. First we develop a dynamic programming method for computing the

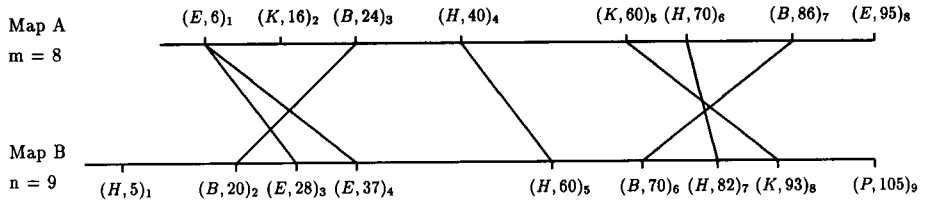

Fig. 2. An illustration of a map alignment.

maximum score. For a segment pair $(i,j;k,l)$, define $F(i,j;k,l)$ to be the score of an optimal global alignment with the rightmost segment pair $(i,j;k,l)$, excluding the term $\mu|(q_m - q_k) - (r_n - r_l)|$ that accounts for the discrepancy between the segment pair $(i,j;k,l)$ and the rightmost ends of the maps A and B. Let $S(k,l)$ be the maximum of $F(i,j;k,l)$ over all $(i,j;k,l)$'s. The following recurrences are used for computing the matrix S.

$$G(i,j) = \max\{S(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| : g < i \text{ and } h < j\}$$

$$F(i,j;k,l) = \sigma(i,j;k,l) + \max\{-\lambda(m - k + i + n - l + j - 2) - \mu|q_i - r_j|,\ G(i,j) + \lambda(k - i + l - j + 2)\}$$

$$S(k,l) = \max_{(i,j;k,l)} F(i,j;k,l)$$

The score of an optimal global alignment of A and B is

$$\max\{S(k,l) - \mu|(q_m - q_k) - (r_n - r_l)| : 1 \le k \le m \text{ and } 1 \le l \le n\}$$

Examining the recurrences, we find that we just need to compute and save the matrix S; the matrix F is computed implicitly. Figure 3 shows a new dynamic programming algorithm for computing the maximum score, where we assume that $S(k,l) = -\infty$ initially for each (k,l).

The time complexity of the algorithm can be obtained easily. A total of $O(m^2n^2)$ time is spent in the two nested **for** loops indexed by g and h. The two nested **for** loops indexed by k and l require a total of $O(R)$ iterations, where R is the number of segment pairs. We need not compute d_j and c_i explicitly; for example, we can break the **for** loop indexed by l when $r_l - r_j > \alpha$. So the algorithm takes $O(m^2n^2 + R)$ time. Note that R is usually much smaller than m^2n^2. The time analysis given here is based on the assumption that the function σ is computed in $O(1)$ time. This can be done when a small number of different enzymes are used in a map, which is usually true in practice. Each enzyme is associated with a unique small integer. A simple hashing technique can be used to compute the number of common enzymes and the number of sites with distinct enzymes between a pair of segments in an incremental fashion, so $\sigma(i,j;k,l)$ can be evaluated in $O(1)$ time. The space complexity of the algorithm is $O(mn)$ since the matrix S is saved.

In practice, we may want to use a simple, approximate method to compute the maximum score. Let δ be a positive integer constant. In computing $G(i,j)$, we take the maximum over only those (g,h)'s such that $\max\{1,i - \delta\} \le g < i$ and $\max\{1,j - \delta\} \le h < j$. The approximate method takes $O(\delta^2mn + R)$ time. Since we do not go back very far in computing G, we only need to save the δ most recently computed rows of S, provided S is computed in order of rows. Thus the space requirement of the method is $O(m + \delta \times n)$.

```
score ← −μ(q_m + r_n) − λ(m + n)
for i ← 1 to m do
    for j ← 1 to n do
    { y ← −μ(q_m + r_n) − λ(m + n)
      for g ← 1 to i−1 do
          for h ← 1 to j−1 do
              y ← max{y, S(g,h) − μ|(q_i − q_g) − (r_j − r_h)|}
      c_i ← max{k : k ≤ m and q_k − q_i ≤ α}
      d_j ← max{l : l ≤ n and r_l − r_j ≤ α}
      for k ← i to c_i do
          for l ← j to d_j do
          { t ← max{−λ(m − k + i + n − l + j − 2) − μ|q_i − r_j|,
                    y + λ(k − i + l − j + 2)}
            S(k,l) ← max{S(k,l), t + σ(i,j;k,l)}
          }
      score ← max{score, S(i,j) − μ|(q_m − q_i) − (r_n − r_j)|}
    }
```

Fig. 3. A new dynamic programming algorithm.

Construction of an optimal alignment. An optimal alignment of the two maps can be found in several ways. If only the matrix S is saved, then the use of a straightforward traceback method allows the alignment to be computed in the same order of time as required by the forward computation. Alternatively, the traceback computation can be done in $O(m + n)$ time at the cost of using additional matrices. In particular, for each (k,l) we need to save (i,j) such that $F(i,j;k,l) = S(k,l)$ and (g,h) such that

$$F(i,j;k,l) = S(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| + \lambda(k - i + l - j + 2) + \sigma(i,j;k,l)$$

Usually $k - i$ and $l - j$ are very small integers, while g and h are quite large. It is possible to pack g and $k - i$, and h and $l - j$ into two single integer words. This method requires two $m \times n$ integer matrices. One disadvantage of the method is that the matrix S has to be saved, which takes $O(mn)$ space.

As discussed before, the approximate method takes $O(m + \delta \times n)$ space to compute the maximum score of those alignments satisfying the restriction that any two adjacent aligned

segment pairs are at most δ sites apart in each map. We can also find an alignment of the maximum score in roughly the same order of space. Hirschberg (1975) designed an elegant divide−conquer algorithm that constructs a longest common subsequence of two sequences of lengths m and n in $O(m + n)$ space. Later Myers and Miller (1988) applied the Hirschberg technique to construction of an optimal global alignment of two sequences under affine gap penalties in $O(m + n)$ space. In the remainder of this subsection, we develop an application of the divide−conquer technique to map comparison. First we briefly review this technique in the context of sequence comparison. Each global alignment of the two sequences corresponds to a path from the point (0,0) to the point (m,n) in a $(m + 1) \times (n + 1)$ grid graph. Finding an optimal alignment means finding a path with the maximum score in the graph. Let $i = \lceil m/2 \rceil$ be the middle row of the graph. An important observation is that each path from (0,0) to (m,n) must intersect row i. In the Hirschberg method, the intersection point (i,j) of an optimal path with row i is first determined, and the two smaller optimal paths from (0,0) to (i,j) and from (i,j) to (m,n) are constructed recursively.

We now apply the Hirschberg method to computation of a largest scoring alignment $(i_1,j_1;k_1,l_1) \ldots (i_d,j_d;k_d,l_d)$ that satisfies $i_t - k_{t-1} \leq \delta$ and $j_t - l_{t-1} \leq \delta$ for each t. (Recall that this restriction is imposed in order to reduce computation time.) Let β be equal to $2\delta + 1$ plus the maximum number of sites in any segment of map A. Take an interval in the center of A from sites x to y with $y - x + 1 = \beta$, assuming that A has more than β sites. First we look at how an alignment intersects this central interval. It can be proved that for any alignment $\Pi = (i_1,j_1;k_1,l_1) \ldots (i_d,j_d;k_d,l_d)$ satisfying the restriction, at least one of the following three equations holds.

$$k_d \leq y \tag{1}$$

$$i_1 \geq x \tag{2}$$

$$k_{t-1} \geq x \text{ and } i_t \leq y \text{ for some } t,\ 1 < t \leq d \tag{3}$$

Equation (1) says that the rightmost pair of Π contains only the site of A that are in or to the left of the central interval. If neither (1) nor (2) holds for the alignment Π, then Π must have two adjacent pairs that both contain some sites from the central interval (equation (3)). For $p = 1,2,3$, let group p contain those alignments that satisfy equation (p).

Constructing an optimal alignment by the Hirschberg method involves computation of dynamic programming matrices from both ends of the maps. Recall that the matrix S is computed in increasing order of site indices. Now we describe the recurrences for the computation in decreasing order of site indices. For a segment pair $(i,j;k,l)$, let $D(i,j;k,l)$ be the score of an optimal global alignment with the leftmost segment pair $(i,j;k,l)$, excluding the discrepancy $\mu|q_i - r_j|$ between the left map ends and $(i,j;k,l)$. Let $C(i,j)$ be the maximum of $D(i,j;k,l)$ over all $(i,j;k,l)$'s. The following recurrences are used for computing the matrix C.

$$E(k,l) = \max\{C(g,h) - \mu|(q_k - q_g) - (r_l - r_h)| : g > k \text{ and } h > l\}$$

$$D(i,j;k,l) = \sigma(i,j;k,l) + \max\{-\lambda(m - k + i + n - l + j - 2) - \mu|(q_m - q_k) - (r_n - r_l)|, E(k,l) + \lambda(k - i + l - j + 2)\}$$

$$C(i,j) = \max_{(i,j;k,l)} D(i,j;k,l)$$

The three equations below give the maximum score for each of the three groups of alignments. An optimal alignment is in the group with the largest overall score, $M = \max\{M_1,M_2,M_3\}$.

$$M_1 = \max\{S(k,l) - \mu|(q_m - q_k) - (r_n - r_l)| : 1 \leq k \leq y \text{ and } 1 \leq l \leq n\} \tag{4}$$

$$M_2 = \max\{C(i,j) - \mu|(q_i - r_j| : x \leq i \leq m \text{ and } 1 \leq j \leq n\} \tag{5}$$

$$M_3 = \max_{\substack{x \leq k < i \leq y, i-k \leq \delta \\ 1 \leq l < j \leq n, j-l \leq \delta}} \{S(k,l) + C(i,j) + \lambda(m + n) - \mu|(q_i - q_k) - (r_j - r_l)|\} \tag{6}$$

Equations (4) and (5) are obvious and correspond directly to (1) and (2). Equation (6) is a little involved. In $S(k,l)$, the sites to the right of site k on A and the sites to the right of site l on B are charged $-\lambda$ per site, as are the sites to the left of site i and j on A and B in $C(i,j)$. So in $S(k,l) + C(i,j)$, the sum of the two penalty terms is $-\lambda(m + n + (i - k + j - l - 2))$. By adding $\lambda(m + n)$ to $S(i,j) + C(k,l)$, we avoid penalizing deleted sites more than once.

We are ready to outline a divide-and-conquer algorithm for constructing an optimal map alignment. If A contains at most β sites, then construct the optimal alignment directly by the traceback procedure as described in the beginning of the discussion on alignment construction. Otherwise, compute $S(k,l)$ for $1 \leq k \leq y$ and $1 \leq l \leq n$, and $C(i,j)$ for $x \leq i \leq m$ and $1 \leq j \leq n$. Then find the entry/entries attaining the largest overall score according to equations (4)−(6). If the largest overall score M is equal to M_3 that is achieved at two entries, say, (k,l) and (i,j), then recursively construct an optimal alignment of $A[1,k]$ and $B[1,l]$ and an optimal alignment of $A[i,m]$ and $B[j,n]$. Note that when computing an optimal alignment of $A[i,m]$ and $B[j,n]$, we no longer charge the distance discrepancy $|q_i - r_j|$ since it has already been taken care of in the previous computation. In case that M is equal to M_1 (or M_2) obtained at (k,l) (or (i,j)), we just apply the procedure recursively to $A[1,k]$ and $B[1,l]$ (or to $A[i,m]$ and $B[j,m]$).

Obviously the algorithm requires $O(\beta(m + n))$ space. We want to show that the algorithm takes $O(\beta^2 mn)$ time. Let $T(m,n)$ denote the time for the algorithm to compute an optimal alignment of two maps of m and n sites respectively. For the analysis, assume that m and β are powers of 2. The algorithm takes the most time when equation (6) gives the largest overall score because it must compute recursively two alignments. It

is easy to see that the total time spent by the algorithm to determine the largest scoring pair of entries, excluding the time for the recursive calls, is bounded by $O(\beta^2 mn + R)$. Since the number R of segment pairs is most $\beta^2 mn$ from the definition of β, the total non-recursive time is $O(\beta^2 mn)$. Thus we have, for some constant c, and for parameters ζ and n', $|\zeta| \leq \beta/2$ and $1 \leq n' < n$, that

$$T(m,n) \leq c\beta^2 mn + T(m/2-\zeta,n') + T(m/2 + \zeta,n - n')$$

where $T(m/2 - \zeta,n')$ and $T(m/2 + \zeta,n - n')$ are the time spent for the recursive calls. We prove by induction on m that $T(m,n) \leq 4c\beta^2 mn$ for $m \geq 2\beta$. Assume that

$$T(m/2 - \zeta,n') \leq 4c\beta^2(m/2 - \zeta)n'$$
$$T(m/2 + \zeta,n - n') \leq 4c\beta^2(m/2 + \zeta)(n - n')$$

Since $1 \leq n' < n$, we obtain $|n - 2n'| < n$. Combining $|\zeta| \leq \beta/2$ and $2\beta \leq m$, we obtain $|\zeta| \leq m/4$. Thus we have that

$$\begin{aligned} T(m,n) &\leq c\beta^2 mn + 4c\beta^2(m/2 - \zeta)n' + \\ &\quad 4c\beta^2\ (m/2 + \zeta)(n - n') \\ &= c\beta^2 mn + 4c\beta^2(mn/2) + 4c\beta^2\zeta(n - 2n') \\ &\leq c\beta^2 mn + 2c\beta^2 mn + 4c\beta^2|\zeta|\ |n - 2n'| \\ &\leq c\beta^2 mn + 2c\beta^2 mn + 4c\beta^2(mn/4) \\ &= 4c\beta^2 mn \end{aligned}$$

Local similarities

First we define best local alignments of two restriction maps and present an algorithm for computing the best local map alignments. Then we discuss a fast approximate method for finding the best local alignments. These algorithms are useful to find unknown repeats in or between maps.

K best local map alignments. A local alignment Π of A and B is a sequence of ordered matching pairs of segments $(i_1,j_1;k_1,l_1)\ (i_2,j_2;k_2,l_2) \ldots (i_d,j_d;k_d,l_d)$, where $k_t < i_{t+1}$ and $l_t < j_{t+1}$ for each $t < d$. Let σ, λ, μ and ν be the same as above. Then the score of the alignment Π is defined to be

$$\text{score}(\Pi) =$$
$$\sum_{t=1}^{d} \sigma(i_t,j_t;k_t,l_t) - \mu \sum_{t=2}^{d} |(q_{i_t} - q_{k_{t-1}}) - (r_{j_t} - r_{l_{t-1}})| -$$
$$\lambda\left[\sum_{t=2}^{d} (i_t - k_{t-1} - 1) + \sum_{t=2}^{d} (j_t - l_{t-1} - 1)\right]$$

In this definition, the regions to the left and right of Π are no longer penalized.

A best local alignment has the maximum score. The recurrences for computing a best local alignment can be developed similarly. For a segment pair $(i,j;k,l)$, define $I(i,j;k,l)$ to be the score of a best local alignment with the rightmost segment pair $(i,j;k,l)$ minus $\lambda(m - k + n - l)$, the penalty for the indels of the sites to the right of sites k and l respectively. Define $H(k,l)$ to be the maximum of $I(i,j;k,l)$ over all $(i,j;k;l)$'s. The following recurrences are used for computing the matrix H.

$$\begin{aligned} j(i,j) &= \max\{H(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| : \\ &\qquad g < i \text{ and } h < j\} \\ I(i,j;k,l) &= \sigma(i,j;k,l) + \max\{-\lambda(m - k + n - l), \\ &\qquad J(i,j) + \lambda(k - i + l - j + 2)\} \\ H(k,l) &= \max_{(i,j;k,l)} I(i,j;k,l) \end{aligned}$$

The score of a best local alignment of A and B is

$$\max\{H(k,l) + \lambda(m - k + n - l) : 1 \leq k \leq m \text{ and } 1 \leq l \leq n\}$$

It is easy to see that a best local alignment can be computed in $O(m^2n^2 + R)$ time and $O(mn)$ space.

Two maps may contain several similar regions of interest to biologists. It is desirable to be able to compute more than one best local map alignment. Waterman and Eggert (1987) introduced the notion of the K best non-intersecting local alignments between two sequences and developed a time-efficient algorithm to compute those K best alignments. Below we develop a method for computing the K best non-intersecting local alignments between two maps by generalizing the Waterman–Eggert technique from sequence comparison to map comparison.

Two segment pairs $(i_1,j_1;k_1,l_1)$ and $(i_2,j_2;k_2,l_2)$ intersect if there is an entry (g,h) such that $\max\{i_1,i_2\} \leq g \leq \min\{k_1,k_2\}$ and $\max\{j_1,j_2\} \leq h \leq \min\{l_1,l_2\}$. Two local map alignments are non-intersecting if no segment pair of one alignment intersects any segment pair of the other alignment. We are interested in computing the K best non-intersecting local map alignments: the largest scoring alignment, then the next largest scoring alignment that does not intersect those already computed, and so on.

For two entries (g,h) and (k,l) of the matrix H with $g < k$ and $h < l$, we say that (g,h) affects (k,l) with respect to H if we have the following equation for some segment pair $(i,j;k,l)$, $i > g$ and $j > h$:

$$H(k,l) = H(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| + \sigma(i,j;k,l) + \lambda(k - i + l - j + 2)$$

For each entry (g,h), we want to determine a region of entries that might be affected by (g,h). A simple way of representing the region is to use the indices of the furthest row and column affected. So if (g,h) affects no entry, then define $row(g,h) = g$ and $col(g,h) = h$. Otherwise, define $row(g,h)$ to be the largest $k > g$ such that (k,t) is affected by (g,h) for some $t > h$, and $col(g,h)$ to be the largest $l > h$ such that (s,l) is affected by

(g,h) for some $s > g$. It is obvious that the matrices *row* and *col* can be computed easily along with the matrix H.

We are ready to present a method for computing the K best non-intersecting local map alignments. Initially the matrices H, *row* and *col* are computed and saved. An entry of H with the maximum score is located and the largest scoring alignment ending at this entry is constructed by a traceback procedure. This alignment is the best local alignment. Assumed that $(i_1,j_1;k_1,l_1)$ and $(i_d,j_d;k_d,l_d)$ are the first and last segment pairs of the alignment. Then recompute the portion of H from rows i_1 to k_d and from columns j_1 to l_d using only segment pairs that do not intersect any segment pair of the best alignment. During the recomputation, parts of the matrices *row* and *col* are modified in such a way that the value for each entry can only be increased. Continue the calculation by adding one row or one column at a time. Let row k and column l be the last row and column of the recomputed region, and let H' denote the new matrix generated in the recomputation process, where $H'(g,h) = H(g,h)$ for each (g,h), $g < i_1$ or $h < j_1$. If there exists some i and j, $i_d < i < k$ and $j_d < j < l$, such that equations (7)–(9) hold

$$H'(g,h) = H(g,h) \text{ for each } (g,h), i \le g \le k \text{ and } j_1 \le h \le l \tag{7}$$

$$H'(g,h) = H(g,h) \text{ for each } (g,h), i_1 \le g \le k \text{ and } j \le h \le l \tag{8}$$

$$row(g,h) \le k \text{ and } col(g,h) \le l \text{ for each } (g,h), i_1 \le g < i \text{ and } j_1 \le h < j \tag{9}$$

then terminate the recomputation process. Next find an entry of H' with the maximum score and report the corresponding alignment, which is the second best non-intersecting alignment. Repeat this process until the K best non-intersecting local alignments are found.

We now examine equations (7)–(9) in detail. In words, equation (7) is to assure that the H' values in the submatrix from rows i to k and from columns j_1 to l are the same as the corresponding H values, and equation (8) is to guarantee that the H' values in the submatrix from rows i_1 to k and from columns j to l are the same as the corresponding H values. Equations (9) implies that the entries in the intersection of rows i_1 through $i-1$ and columns j_1 through $j-1$ cannot affect any entry beyond row k or column l. Let H' be the matrix calculated if the recomputation is carried out to row m and column n. We want to show that there is no need to carry out the recomputation to the end, i.e. $H'(s,t) = H(s,t)$ for each (s,t), $s > k$ and $t \ge j_1$, or $s \ge i_1$ and $t > l$. Since we cannot use the segment pairs of the already determined alignments in computing H', we have $H'(g,h) \le H(g,h)$ for each (g,h) $g \ge i_1$ and $h \ge j_1$. Take an entry (s,t), $s > k$ and $t \ge j_1$, or $s \ge i_1$ and $t > l$. Assume that (s,t) is affected by some (s',t'), $s' \le k$ and $t' \le l$, with respect to H. By (9) we conclude that at least one of the following cases is true: (i) $s' < i_1$, (ii) $t' < j_1$, (iii) $i \le s' \le k$ and $j_1 \le t' \le l$ or (iv) $i_1 \le s' \le k$ and $j \le t' \le l$. It follows from the definition of H' and by equations (7) and (8) that $H'(s',t') = H(s',t')$. Since (s',t') affects (s,t), we have, for some segment pair $(i,j;s,t)$, $i > s'$ and $j > t'$,

$$\begin{aligned} H(s,t) &= H(s',t') - \mu|(q_i - q_{s'}) - (r_j - r_{t'})| + \sigma(i,j;s,t) + \lambda(s - i + t - j + 2) \\ &= H'(s',t') - \mu|(q_i - q_{s'}) - (r_j - r_{t'})| + \sigma(i,j;s,t) + \lambda(s - i + t - j + 2) \\ &\le H'(s,t) \end{aligned}$$

Thus $H'(s,t) = H(s,t)$.

The time complexity of the algorithm is

$$O(m^2n^2 + mn \sum_{t=1}^{K-1} Q_t)$$

where $O(m^2n^2)$ time is spent in the initial complete sweep and Q_t is the number of entries recomputed after the tth best alignment is found. Note that it takes $O(mn)$ time to compute one entry by the naive dynamic programming method.

An approximate method. The algorithm discussed above is expensive in time and space. Here we give an approximate method that is easy to implement. This method also has its counterpart in sequence comparison. Gotoh (1987) developed an approximate algorithm for comptuing K local alignments between two sequences. The K alignments produced by the method are not necessarily the K best non-intersecting alignments. In this method, alignments with the same left endpoint are put into the same group and the right endpoint of the largest scoring alignment in each group is saved. In a single sweep of the dynamic programming matrix, the left and right endpoints of the largest scoring alignment in each of the K best groups are determined. Then each of these K alignments is constructed. The single sweep can be made in space proportional to the sequence lengths. Although the construction of each alignment usually requires a limited amount of space, its worst-case complexity is quadratic in the alignment length. Huang *et al.* (1990) observed that each local alignment can be constructed in linear space by the method of Myers and Miller (1988).

Consider the extension of the Gotoh algorithm to map comparison. For a local alignment $\Pi = (i_1,j_1;k_1,l_1) \ldots (i_d,j_d;k_d,l_d)$, (i_1,j_1) and (k_d,l_d) are called the left and right endpoints of Π respectively. For an entry (g,h), define $L(g,h)$ to be the left point of an alignment ending at (g,h) of score $H(g,h)$. The matrix L is easily computed together with H.

Now we describe the method. The first step is to obtain the K largest scoring entries (g_1,h_1), (g_2,h_2), $\ldots$, (g_K,h_K) with $L(g_s,h_s) \ne L(g_t,h_t)$, $s \ne t$, by computing the matrices H and L in a single sweep. Then construct the alignment with the

endpoints $L(g_s,h_s)$ and (g_s,h_s) for each s, $1 \leq s \leq K$, using the space-efficient procedure as discussed in the 'Global similarities' subsection. To speed up the sweep, we just use the δ^2 entries (g,h), $i - g \leq \delta$ and $j - h \leq \delta$, to calculate $H(i,j)$, which requires that we save at most the δ most recent rows of H. It should be pointed out that the map alignments produced by this approximate method may be intersecting. However, this rarely happens in practice, as will be demonstrated in the Implementation section.

The initial sweep takes $O(\beta^2 mn)$ time and $O(\beta(m + n) + K)$ space, where $O(K)$ space is devoted to the storage of the K best entires along with their L values. Observe that β is equal to $2\delta + \max\{k - i + 1 : 0 \leq q_k - q_i \leq \alpha\}$. The construction of an alignment of length L_1 by the space-efficient method takes $O(\beta^2 L_1^2)$ time and $O(\beta L_1)$ space. Therefore the approximate method requires

$$O\left(\beta^2 mn + \beta^2 \sum_{t=1}^{K} L_t^2\right)$$

time, where L_t is the length of the tth alignment. Its space requirement is $O(\beta(m + n) + K)$.

Faster algorithms

Myers and Huang (1992) presented a time-efficient algorithm for computing the minimum distance between two restriction maps under the basic model proposed by Waterman *et al.* (1984). The algorithm takes $O(mn(\log m + \log n))$ time and $O(mn)$ space. The description of the algorithm was given in the context of the distance measure, where minimum operations are performed. Now we briefly describe the Myers–Huang algorithm in the context of the similarity measure. Consider the recurrences for computing the matrix X as given in the introduction.

$$Y(i,j) = \max\{X(g,h) - \mu|(q_i - q_g) - (r_j - r_h)|: g < i, h < j \text{ and } a_g = b_h\}$$

$$X(i,j) = \max\{v - \lambda(m + n - 2) - \mu|q_i - r_j|, Y(i,j) + v + 2\lambda\}$$

To compute each $Y(i,j)$ efficiently, a list of computed entries with their X-values is kept. If a computed entry could not possibly affect any entry to be calculated, then the computed entry (called a dead entry) with its X-value is removed from the list. Newly computed entries are placed into the list. Assume that the matrix X is computed in increasing order of i. After removal of dead entries, each entry (g,h) in the list is associated with a sub-interval of the interval $[0,r_n]$ in such a way that (g,h) affects those entries (i,j) with r_j in that sub-interval. These sub-intervals form a partition of the interval $[0,r_n]$. The entries in the list are ordered linearly according to the increasing order of the associated sub-intervals. The list is implemented by a balanced search tree so that the value $X(g,h)$ yielding the maximum score $Y(i,j)$ can be located in $O(\log m + \log n)$ time, instead of $O(mn)$ time. Thus the algorithm takes $O(mn(\log m + \log n))$ time to compute the matrix X. If the matrix X and the trace-back information are saved, then $O(mn)$ space is required.

Now we consider the application of the Myers–Huang technique to the computation of global and local alignments under the new model of map comparison. First look at the recurrences involving S,F and G for computing a single optimal alignment. The most costly computation is:

$$G(i,j) = \max\{S(g,h) - \mu|(q_i - q_g) - (r_j - r_h)| : g < i \text{ and } h < j\}$$

which requires $O(mn)$ time by the straightforward method. This recurrence is essentially the one that was treated by Myers and Huang (1992), which implies that $G(i,j)$ can be computed in $O(\log m + \log n)$ time. Thus an optimal global alignment can be found in $O(mn(\log m + \log n) + R)$ time and $O(mn)$ space.

With the same observation, we see that the best local map alignment can be computed in $O(mn(\log m + \log n) + R)$ time and $O(mn)$ space, and the K best non-intersecting local map alignments can be computed in $O(Kmn(\log m + \log n) + KR)$

Table I. Local alignments within the *E.coli* map

Number	Score	Region 1	Region 2
1	76	2055.9–2060.1	3474.1–3479.8
2	68	121.3–126.1	2093.0–2097.4
3	64	3466.9–3469.4	3592.4–3594.7
4	62	292.2–295.0	3781.8–3784.4
5	61	2119.7–2124.6	2265.6–2270.1
6	60	2271.6–2273.8	3676.4–2678.9
7	60	987.0–990.6	3656.9–3661.2
8	59	2161.1–2164.2	3676.4–3679.3
9	59	916.0–917.9	4012.8–4015.1
10	59	229.7–233.0	3776.3–3780.2
11	58	2041.6–2043.6	3552.8–3554.8
12	58	3462.4–3469.0	3581.7–3587.3
13	58	3521.3–3527.1	3788.3–3794.6
14	57	2039.2–2044.7	3058.4–3062.9
15	57	1814.2–1818.2	2652.2–2656.4
16	54	190.0–192.4	1856.4–1859.0
17	54	229.7–230.9	2921.7–2923.2
18	54	315.4–316.6	714.9–716.4

Table II. Multiple similar regions within the *E.coli* map

Group	Copy 1	Copy 2	Copy 3	Copy 4	Copy 5	Copy 6
1	2271.6	2373.4	2814.4	3546.5	3676.4	3953.4
2	916.0	1289.7	2040.9	2573.7	3677.6	4012.9
3	738.2	2040.5	2162.7	3953.7	4232.8	
4	229.4	2488.2	2921.7	4234.8		
5	315.4	714.9	4674.5			
6	2573.0	2724.1	4012.8			
7	715.9	2162.7	3552.8			
8	2040.5	2573.7	4098.9			
9	988.0	2197.8	3658.1			
10	1581.5	2755.4	2925.7			

time and O(mn) space by making K complete passes of the matrix.

It should be pointed out that the algorithm of Myers and Huang (1992) requires that each map is strictly increasing in site position, i.e. $q_i < q_{i+1}$ and $r_j < r_{j+1}$ for each i and j. If map data do not satisfy this requirement, we can modify the map data slightly so that they satisfy the requirement. Of course, we have to set α to a value big enough for a segment to cover the map sites that are originally at the same position.

Huang and Miller (1991) developed a space-efficient version of the algorithm of Waterman and Eggert (1987) for computing the K best non-intersecting local alignments between two sequences. This space-efficient technique, combined with the time-efficient method of Myers and Huang (1992), has been applied to the construction of map alignments under the previous model (K.Chao and W.Miller, personal communication). We think that those techniques are also applicable to our new model of map comparison, though the resulting algorithm might be very complicated.

Implementation

We have implemented as portable C programs two approximate algorithms: the space-efficient one for aligning two entire maps described in the 'Global similarities' subsection, and the space-efficient one for finding local similarities between two maps presented in the 'Local similarities' subsection. The programs are used to analyze a restriction map of the *E.coli* genome constructed by Kohara *et al.* (1987). The *E.coli* map contains >7000 sites for eight different restriction enzymes. Because of the limited gel resolution, the map is subject to the two types of errors we have discussed: the incorrect order of closely spaced sites for distinct enzymes and the merge of several sites for the same enzyme into a single site. Observe that the analysis of the *E.coli* map requires the use of space-efficient algorithms on conventional computers.

To find repeated regions within a single map, the local similarity algorithm has to be modified so that the computation is restricted to the upper triangular part of the matrix. A pair of segments from the same map can be aligned only if they have no site in common. So the trivial alignment of some map region with itself is excluded.

There are two digitized versions of Kohara's map: one made by D.L.Daniels of University of Wisconsin, and the other by K.E.Rudd of Food and Drug Administration. The Daniels map was used in a statistical analysis of the distribution of restriction sites (Churchill *et al.*, 1990), while the Rudd map was utilized in the localization of sequenced genes on the genome (Rudd *et al.*, 1990,1991). To compare the two maps, the numbers showing site positions in base pairs on the Daniels map were scaled to be in 100 base units, the scale for site positions of the Rudd map.

Consider choosing the values for the parameters ν, μ, λ, α and δ. It is observed that the errors usually occur in a region <0.5 kb in length (Kohara *et al.*, 1987). So we set α to 5 since site positions are in 100 bp. We always set ν to 10. The selection of the values for μ and λ depends on experience. We found that the choice of $\mu = 2$ and $\lambda = 5$ works well in practice. It should be pointed out that the magnitude of the values for these parameters is determined by the scale for site positions of map data. The parameter δ is an upper bound on the number of sites allowed between every pair of adjacent segments in a computed alignment. The choice of $\delta = 10$ seems to allow interesting alignments to be computed in a reasonable time.

The global map alignment program was used to compare the two maps. Generally, the Rudd map has more sites than the Daniels map. Therefore, the optimal alignment shows the locations of insertion of sites into the Daniels map (or deletion of sites from the Rudd map). In addition, the optimal alignment shows the locations where closely spaced sites are ordered differently between the two maps.

The local map alignment program was applied to each map to detect similar regions on the map, where the program generates local alignments in order to score. In both maps, the top alignments involve the four regions that are known to be copies of an insertion sequence (Muramatsu *et al.*, 1988). The highest scoring alignment from the Daniels map shows fewer sites that are out of order than the corresponding one from the Rudd map. For reason of space, below we report only the results obtained with the Daniels map. The program was used to compute 100 largest scoring local alignments. We rarely found any intersecting alignments in the 100 alignments. The four copies of the insertion element form six pairs of similar regions, all of which show up in the top five alignments with one of the alignments containing two pairs. Those five alignments show very strong matches. Although the remaining alignments are short and of low score, some of them might be of interest and are listed in Table I. Table I shows 18 alignments with one alignment per row. The column 'score' lists the score of each alignment. The locations of aligned regions in kb are given in the columns 'Region 1' and 'Region 2'. The experimental results show that the local alignment program performs well at producing non-intersecting, high-scoring alignments in practice.

It is not easy to identify multiple copies of a short repeated element from the output of the program. To facilitate this task, we developed a search program that takes every region of a fixed number l of sites called a probe and searches the entire map for all approximate occurrences of the probe. The program was run with $l = 7, 8, 9$. The results are presented in Table II. Here 10 groups of similar regions are shown with one group per row. Only the starting location of each region in a group is given, where the numbers showing the locations are in kb.

Discussion

We did not present a specific algorithm for finding the best overlap between two map fragments—a problem that is of interest in physical mapping (e.g. Kohara *et al.*, 1987). In the

sequence case, a variant of the Smith–Waterman algorithm can be used to compute the overlapping alignment of the maximum score between two sequence fragments in quadratic time (X.Huang, submitted). An algorithm for detecting the best overlap between two map fragments can easily be obtained by modifying the local map similarity algorithm presented in this paper.

We have generalized the previous model of map comparison to address two types of map errors. The new model still allows dynamic programming techniques to compute an optimal map alignment. We have also demonstrated that many efficient techniques employed in sequence comparison and map comparison under the previous model are applicable to map comparison under the new model. It is expected that the algorithms presented in this paper will be useful as more and more restriction map data become available. Readers interested in the computer programs may contact the authors by electronic mail at huang@cs.mtu.edu or msw@hto.usc.edu.

Acknowledgements

We thank Donna Daniels and Ken Rudd for kindly providing the map data. We also thank the referees for suggestions. The work was done when X.H. was visiting USC. This research was supported by the National Science Foundation and the National Institutes of Health.

References

Churchill,G.A., Daniels,D.L. and Waterman,M.S. (1990) The distribution of restriction enzymes sites in *Escherichia coli*. *Nucleic Acids Res.*, **18**, 589–597.

Coulson,A., Sulston,J., Brenner,S. and Karn,J. (1986) Toward a physical map of the genome of the nematode *Caenorhabditis elegans*. *Proc. Natl. Acad. Sci. USA*, **83**, 7821–7825.

Gotoh,O. (1987) Pattern matching of biological sequences with limited storage. *Comput. Applic. Biosci.*, **3**, 17–20.

Hirschberg,D.S. (1975) A linear space algorithm for computing maximal common subsequences. *Commun. ACM*, **18**, 341–343.

Huang,X., Hardison,R.C. and Miller,W. (1990) A space-efficient algorithm for local similarities. *Comput. Applic. Biosci.*, **6**, 373–381.

Huang,X. and Miller,W. (1991) A time-efficient, linear-space local similarity algorithm. *Adv. Appl. Math.*, **12**, 337–357.

Kohara,Y., Akiyama,K. and Isono,K. (1987) The physical map of the whole *E.coli* chromosome: application of a new strategy for rapid analysis and sorting of a large genomic library. *Cell*, **50**, 495–508.

Miller,W., Ostell,J. and Rudd,K.E. (1990) An algorithm for searching restrictions maps. *Comput. Applic. Biosci.*, **6**, 247–252.

Miller,W., Barr,J.C. and Rudd,K.E. (1991) Improved algorithms for searching restriction maps. *Comput. Applic. Biosci.*, **7**, 447–456.

Muramatsu,S., Kato,M., Kohara,Y. and Mizuno,T. (1988) Insertion sequence IS5 contains a sharply curved DNA structure at its terminus. *Mol. Gen. Genet.*, **214**, 433–438.

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

Myers,E.W. and Huang,X. (1992) An $O(N^2 \log N)$ restriction map comparison and search algorithm. *Bull. Math. Biol.*, in press.

Olson,M.V., Dutchik,J.E., Graham,M.Y., Brodeur,G.M., Helms,C., Frank,M., MacCollin,M., Scheinman,R. and Frank,T. (1986) Random-clone strategy for genomic restriction mapping in yeast. *Proc. Natl. Acad. Sci. USA*, **83**, 7826–7830.

Rudd,K.E., Miller,W., Ostell,J. and Benson,D.A. (1990) Alignment of *Escherichia coli* K12 DNA sequences to a genomic restriction map. *Nucleic Acids Res.*, **18**, 313–321.

Rudd,K.E., Miller,W., Werner,C., Ostell,J., Tolstoshev,C. and Satterfield,S.G. (1991) Mapping sequenced *E.coli* genes by computer: software, strategies and examples. *Nucleic Acids Res.*, **19**, 637–647.

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

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

Waterman,M.S. and Raymond,R. (1987) The match game: new stratigraphic correlation algorithmns. *Math. Geol.*, **19**, 109–127.

Waterman,M.S., Smith,T.F. and Katcher,H.L. (1984) Algorithms for restriction map comparisons. *Nucleic Acids Res.*, **12**, 237–242.

Received on January 6, 1992; accepted on March 10, 1992