Further improvement in methods of
group-to-group sequence alignment
with generalized profile operations

Osamu Gotoh

Abstract

It has previously been shown that rigorous optimization of alignment between two groups of sequences in the sense of minimal sum of pairs (SP) score with a linear gap-weighting function can be achieved by an extended version of the dynamic programming algorithm. The major drawback of this algorithm was that the computation time grows in proportion to the product of the numbers (M and N) of sequences comprising the two groups. A new algorithm presented in this paper achieves the same rigorous alignment in a time complexity much less dependent on the sizes of the groups. Examinations with many groups of sequences indicated that the new algorithm runs faster than the earlier one when \(M \times N > 6-10\), \(\approx 10\) times faster when \(M \times N \approx 200\), and > 100 times faster when \(M \times N > 2500\). This computational acceleration facilitates application of the algorithm to alignment of large groups, especially in the framework of iterative refinement strategies.

Introduction

It has been expected and evidenced that alignment of distantly related groups of pre-aligned homologous sequences generally provides more accurate results than that between single individual members (Argos, 1987; Gotoh, 1992). Group-to-group alignment is also an integral step in most iterative multiple sequence alignment methods (Chan et al., 1992). It is currently most popular to assess the goodness of a multiple sequence alignment by the so-called ‘sum of pairs’ or SP measure (Carrillo and Lipman, 1988) with a linear gap-weighting function of the form of \(w(k) = uk + v\), where \(k\) is the length of the gap, and \(v\) and \(u\) are costs assigned to opening and extension of a gap respectively. Recent analysis of insertions/deletions in protein tertiary structures has suggested a major contribution of the gap-opening cost \(v\) to \(w(k)\) (Pascarella and Argos, 1992). Although an efficient dynamic programming algorithm for aligning two sequences with a linear gap-weighting function has long been known (Gotoh, 1982), a rigorous algorithm that optimizes alignment between two groups of sequences with internal gaps has only recently been developed (Gotoh, 1993). However, this rigorous algorithm (Algorithm D) and the related, slightly less accurate algorithm (Algorithm C) had a drawback in that they took computation time in proportion to \(M \times N\), where \(M\) and \(N\) are the numbers of the members of the two groups. Because of the rapid accumulation of nucleotide and amino acid sequences in databases, it is not unusual for each group or family to consist of tens or even hundreds of primary sequences. Alignment between such groups by existing methods could be intolerably time consuming. As realized in the approximate algorithms (A) and (B) described previously (Gotoh, 1993), it is possible partly to suppress this computational augmentation by the use of profile vectors (Gribskov et al., 1987). However, there has been no way to evaluate the accurate gap-opening costs with computation steps less than \(O(M \times N)\).

I present here a new algorithm which enables fast and accurate evaluation of the gap costs. An essential point of the algorithm is to separate the pre-existing internal (or ‘static’) gaps from the ‘dynamic’ gaps that are inserted in the alignment process. The status of these two kinds of gaps is separately encoded into sets of static or dynamic gap-state variables with related but distinct record types. The static and dynamic gap-state variables are combined together to generate the full gap states when needed. The static gap state variables are calculated once before the main alignment routine, and hence much of duplicate operations can be eliminated. The computation time is not directly dependent on the numbers of sequences in the two groups but depends on the complexities in distribution of internal gaps. Since this complexity generally increases with the number of sequences in a group, the computation time does so but in a much slower rate than \(M \times N\).

Experiments with many groups of sequences indicated that the new algorithm runs faster than the earlier one when \(M \times N > 6-10\). The same experiments also revealed the conditions under which use of profile vectors is advantageous, though these conditions may be somewhat dependent on machine and implementation. Based on these results, the current alignment programs (alp/aln and rrp/rrn: Gotoh, 1993) are designed to select the most appropriate algorithm at run time.

Department of Biochemistry, Saitama Cancer Center Research Institute, 818 Komuro, Ina-machi, Saitama 362, Japan

© Oxford University Press
system and methods

The programs were written in standard C and run under Unix (4.2 BSD on Sony News 830 or Sun OS4.1.1 on SPARCstation2). Most calculations were made on a Sun workstation equipped with 32 MBytes of memory, but the slower machine was used to measure short CPU time.

Algorithm

SP score between aligned groups

We will first discuss the methods for evaluating the SP score of a given alignment between two groups of sequences. The major features of the new algorithm are demonstrated in this subproblem, while the points relevant to alignment of two groups by dynamic programming algorithms will be discussed in later subsections.

Consider a group of sequences, \( A \), of length \( I \) is composed of \( M \) pre-aligned sequences. Each element of \( A \), \( a_{m,i} \) \( (1 \leq m \leq M, 1 \leq i \leq I) \in X \), is either a character representing an amino acid or nucleotide residue or a null indicating a deletion, and so the alphabet set \( X \) is composed of five (for nucleotide) or 21 (for amino acid) elements. (A deletion in this paper refers to a single site, while a gap is a span of consecutive deletions within a sequence.) Such an internal or 'static' deletion is represented by \( \Delta \) in the following text. A Boolean variable \( q_{m,i} = \Delta \) indicates whether \( a_{m,i} \) represents a deletion or an ordinary residue. Any Boolean variable is assigned the value 1 if true, and 0 if false. The symbols \( \Delta \) and \( * \) indicate deletion and non-deletion sites respectively.

The gap state at site \( (m,i) \), \( Q^A_{m,i} \), is defined as the number of consecutive nulls including, and immediately preceding the element \( a_{m,i} \), along the row \( m \) (Gotoh, 1993). We will represent by \( f^A_{m,i} \) the number of occurrence of element type \( x \in X \) on a column \( i \) of \( A \). The vector \( p^A_i \), of dimension 5 or 21, will be called a frequency vector. The 'profile' (Gribskov et al., 1987) at position \( i \) is represented by another vector, \( p^A_i \), of the same dimension as that of \( f^A_{m,i} \):

\[
p_{x,i} = \sum_{y \in X} d(x,y) \cdot f_{y,i} \tag{1}
\]

where \( d(x,y) \) denotes the measure of dissimilarity between elements \( x \) and \( y \).

Let us calculate the SP score between \( A \) and another group of sequences, \( B \), with \( N \) members and \( I \) columns matched to those of \( A \). Various variables for \( B \) are defined analogously as above with the superscript of 'B' instead of 'A'. From the truth table (Gotoh, 1993) reproduced in Figure 1(A), we can evaluate SP(\( A, B \)) by

\[
\text{SP}(A, B) = \sum_{i=1}^{I} \sum_{m=1}^{M} \sum_{n=1}^{N} \{ d(a_{m,i}, b_{n,i}) + v \cdot g_{m,n,i} \} \tag{2}
\]

where \( v \) is the constant term of a linear gap-weighting function. By the use of profile and frequency vectors, the summation of the first term in equation (2) over \( m \) and \( n \) can be transformed into one of the four ways:

\[
\sum_{m=1}^{M} \sum_{n=1}^{N} d(a_{m,i}, b_{n,i}) = \sum_{n=1}^{N} p^A_{n,i} = \sum_{m=1}^{M} p^B_{m,i} \tag{4a}
\]

\[
= \sum_{x \in X} f^A_{x,i} \cdot p^A_{x,i} = \sum_{x \in X} f^B_{x,i} \cdot p^B_{x,i} \tag{4b}
\]

Depending on the values for \( M \) and \( N \), we can choose the most economical one among these five expressions.
Static gap state variables

Consideration of the above suggests the presence of more efficient methods for computing the summation of the second term in equation (2) by collectively considering those sequences with an identical $Q_{m,i}$ value at a column position $i$. Let us define a record type $SGS = (slen, freq)$. The first member of this record corresponds to the gap state, i.e. the length of the gap up to the current site, and the second member indicates the number of sequences that have this gap state value at a given column position. Thus, a set of SGS-type records $r_{k,i}^A \in R_i^A$ ($1 \leq k \leq |R_i^A|$) defined by equation (5(a)) below is analogous to $f_{k,i}$ in that it represents a categorized feature of the group of sequences at a given column position.

$$r_{k,i}^A \cdot freq \equiv \sum_{m=1}^{M} (Q_{m,i} = r_{k,i}^A \cdot slen) > 0 \quad (5a)$$

The cardinality of this set, $|R_i^A|$, equals the number of distinct $Q_{m,i}$ values found at the column position $i$, and is usually much smaller than $M$ when $M$ is appreciably large. We assume that the members of $R_i^A$, as well as those of other related sets described below, are ordered according to the slen values, i.e. if $r_{k,i}^A \in R_i^A$, $r_{k',i}^A \in R_i^A$ and $k < k'$, then $r_{k,i}^A \cdot slen < r_{k',i}^A \cdot slen$.

As shown later, $R_i^A$ plays rather a subsidiary role in evaluation of gap-opening costs. The leading part is performed by two complementary sets of SGS-type records, $S_i^A$ and $T_i^A$. A record $s_{k,i}^A \in S_i^A$ specifies the number of sequences with $a_{m,i} \neq \Delta$ $(q_{m,i} = 0)$ and a given value of $Q_{m,i-1}$ at the column position $i$, whereas $t_{k,i}^A \in T_i^A$ specifies the corresponding number with $a_{m,i} = \Delta$ $(q_{m,i} = 1)$. Thus, each member of $S_i^A$ and $T_i^A$ is expressed as:

$$s_{k,i}^A \cdot freq \equiv \sum_{m=1}^{M} (1 - q_{m,i}^A) \cdot (Q_{m,i-1} = s_{k,i}^A \cdot slen) > 0 \quad (5b)$$

$$t_{k,i}^A \cdot freq \equiv \sum_{m=1}^{M} q_{m,i}^A \cdot (Q_{m,i-1} = t_{k,i}^A \cdot slen) > 0 \quad (5c)$$

For practical purposes, it is more convenient to use the ‘cumulative’ form of $s_{k,i}^A$, rather than $s_{k,i}^A$ itself:

$$s_{k,i}^{A+} \cdot freq \equiv \sum_{m=1}^{M} (1 - q_{m,i}^A) \cdot (Q_{m,i-1} \geq s_{k,i}^A \cdot slen) \quad (5d)$$

Gap-state variables of the cumulative form will be indicated by a plus sign in the superscript. Figure 1(B) shows an example of a short multiple sequence alignment with values for various associated variables.

The summation of the second term of equation (2) can now be expressed:

$$\sum_{m=1}^{M} \sum_{n=1}^{N} g_{m,n,i} = T_i^A \ast S_i^{A+} + T_i^B \ast S_i^{A'} \quad (6)$$

where the ‘inner product’ denoted by an asterisk is defined as

$$T \ast S^+ = \sum_{i=1}^{T} t \cdot freq \times s^+([t \cdot slen]) \cdot freq \quad (7)$$

Figure 1(C) illustrates the principle of this calculation. $s^+([L])$ denotes the record that has the smallest slen value in the subset $\{s^+ \in S^+ ; s^+ \cdot slen \geq L\}$. If the subset is empty, $s^+([L]) \cdot freq$ is defined to be 0. Arithmetic operations at most $O(|T_i| + |S_i^{A+}|)$ suffice for calculation of the inner product as defined by equation (7). As in the case of equation (4), we can choose either equation (3) or equation (6) to get $\sum_{m=1}^{M} \sum_{n=1}^{N} g_{m,n,i}$ depending on the circumstances. We may call the set of vectors of $f_i$, $p_i$, $S_i^+$ and $T_i$ ‘generalized profile vectors’.

SP score of a single group

The SP score of a single group, $A$, is defined in a similar manner as that between two groups shown by equation (2):

$$SP(A) = \sum_{i=1}^{T} \sum_{m=1}^{M} \sum_{n=1}^{N} \{d(a_{m,i}, a_{n,i}) + v \cdot g_{m,n,i}\} \quad (8)$$

The techniques used in equations (4) and (6) are applicable to transform this expression to profile-based ones with minor modifications:

$$SP(A) = \sum_{i} \left[ \sum_{x \in X} f_{x,i} \left( \sum_{y < x} d(x,y) f_{x,y} \right) + d(x,x)(f_{x,i} - 1)/2 \right] + v \cdot T_i^A \ast S_i^{A+}$$

$$SP(A) = \sum_{i} \left[ \sum_{x \in X} f_{x,i} \{p_{x,i} - d(x,x)/2 + v \cdot T_i^A \ast S_i^{A+} \} \right] \quad (9a)$$

The latter form of equation (9) is valuable only when both frequency and profile vectors of the group have been calculated and are stored in memory.

Modification of algorithm (C)

In a previous report (Gotoh, 1993), I proposed four algorithms, (A)–(D), to obtain an alignment between two groups of sequences. Simpler algorithms (A) and (B) do not, but more advanced algorithms (C) and (D) do accurately evaluate gap-opening costs. Algorithm (C) follows the standard dynamic programming procedure (Needleman and Wunsch, 1970), while algorithm (D) adopts the candidate list paradigm (Miller and Myers, 1989) to achieve rigorous optimization, which may not be attained by algorithm (C) in some cases. Since algorithm
O. Gotoh

(C) is simpler than algorithm (D), I first describe algorithm (C) and its adaptation to use of generalized profiles.

Suppose that we are aligning two groups of sequences, A and B, with M and N members and of lengths of I and J. Each column vector \( a_i \) (1 ≤ i ≤ I) of the dimension of M or \( b_j \) (1 ≤ j ≤ J) of the dimension of N behaves as a unit in the alignment process. According to the standard procedure of dynamic programming with a linear gap-weighting function (Gotoh, 1982), we fill matrices \( D_{i,j}^k \) (1 ≤ α ≤ 3) by the following recursion relations with appropriate initial conditions:

\[
\begin{align*}
D_{i,j}^1 &= \min_{\beta=1,3} \{ D_{i-1,j}^\beta + v \cdot g_{i,j}^\beta \} + d(a_i, -) \quad (10a) \\
D_{i,j}^2 &= \min_{\beta=2,3} \{ D_{i,j-1}^\beta + v \cdot g_{i,j}^\beta \} + d(-, b_j) \quad (10b) \\
D_{i,j}^3 &= \min_{\beta=1} \{ D_{i-1,j-1}^\beta + v \cdot g_{i,j}^\beta \} + d(a_i, b_j) \quad (10c)
\end{align*}
\]

The superscript \( \alpha \) indicates the direction of the last edge in the corresponding path matrix (e.g. Figure 2B in Gotoh, 1993), and \( g_{i,j}^\alpha \) denotes the number of gaps that open in association with this edge (for more details, see Gotoh, 1993). A dash ‘-‘, instead of ‘Δ’, is used to distinguish a ‘dynamic’ deletion (or deletions that span a column) newly introduced in the alignment process from a pre-existing static deletion.

In the earlier algorithm, \( g_{i,j}^{\beta 0} \) was calculated in a similar way to that shown in equation (3) but with the running gap states, \( Q_{m,i,j}^A \) and \( Q_{n,j,j}^B \), which were obtained at each recursion step for individual primary sequences. (We will use an asterisk to distinguish a running variable from the static counterpart.) This took \( O(M \times N) \) operations at each node and heavily limited the overall performance. Since the static part of gap states has already been known, the complete running gap states can be recomputed from the static part combined with the dynamic part which alone is subjected to recursion. At a given node, the static part is common to all the paths, \( \beta \alpha \in \{11, 13, 22, 23, 31, 32, 33\} \), and only the dynamic part is variable. On the other hand, the dynamic part is common to all member sequences in a group. These features help to reduce both computation time and memory required in the recursion process.

One may suppose that the dynamic gap state might be represented by a single variable \( k \), the length of a horizontal or vertical path segment (a run of edges of the same direction) leading to the current node (Figure 2A). This is true if no internal (static) gap is present in either group of sequences. Otherwise, however, an internal gap may ‘channel’ separate dynamic gaps (Figure 2A). Since the channelling occurs as a function

![Fig. 2. (A) Transitions of the dynamic gap states in the recursion process. The path indicated by the solid line at the lower left corner corresponds to the partial alignment shown above, where only one of the groups is presented. Static deletions, dynamic deletions, and ordinary (non-deletion) residues are represented by the symbols of Δ, - and * respectively. The dots indicate the \( k \) consecutive dynamic deletions corresponding to the long horizontal segment in the path. The shaded records are members of the minimal subset sufficient to restore the complete dynamic gap states. (B) The procedures of recursion on the dynamic gap states corresponding to the induction processes I and II shown in (A).]

of the length of the internal gap, the complete information as to the dynamic gap state needs to be held by a slightly more complex structure.

Let us define a second two-member record type, \( \text{DGS} = \{\text{slen}, \text{dlen}\} \), so that each dynamic gap state is encoded by a set of records of this type, \( W_{i,j}^{(d)} \) (0 ≤ i ≤ I, 0 ≤ j ≤ J, \( C \in \{A, B\} \), 1 ≤ \( \alpha \) ≤ 3). We assume that one DGS-type record \( \omega_{k,j}^{(d)} \) in \( W_{i,j}^{(d)} \) is allocated for each SGS-type record \( r_{k,i} \in R_{i,j}^A \) or \( r_{k,j} \in R_{i,j}^B \), where \( R_{i,j}^A \) and \( R_{i,j}^B \) are defined in a previous section. The first member of DGS-type record works as the key to link the corresponding static and dynamic gap state records.

The second member indicates the cumulative number of the preceding and present dynamic deletions uninterupted by an ordinary (non-deletion) residue. At the initial stage, each \( W_{0,j}^{(d)} \) is set to \((0,0)\). Other \( W_{i,j}^{(d)} \) are then generated by induction in association with equations (10). For the example shown in Figure 2(A), the members of \( W_{i,j}^{(d)} \) and those of \( W_{i+1,j}^{(d)} \) are obtained in association...
with equations (10c) and (10a) by the procedures I and II shown in Figure 2(B), respectively. The induction rules for other $W_{i,j}^{\alpha}$ are essentially the same as either of the procedures depending on the last status being deletion or not. The above definition of $W_{i,j}^{\alpha}$ implies that $|W_{i,j}^{\alpha}| = |R_i^\alpha|$ and $|W_{i,j}^{\beta}| = |R_j^\beta|$. In fact, the entire set can be restored from the minimal subset of $W_{i,j}^{\alpha}$ (Figure 2A), and so only a few records are usually sufficient to encode each dynamic gap state.

Now let us construct four running gap states, $S_{i,j}^{\alpha+}$, $S_{i,j}^{\beta+}$, $S_{i,j}^{\alpha+}$, and $S_{i,j}^{\beta+}$, from static and dynamic gap state variables. Once constructed, they are usable to calculate $g_{i,j}^{\beta}$ in just the same way as that shown in equation (6). We introduce an operator $\leftarrow$ to indicate that its left member is the running variable constructed by the combination of static and dynamic variables on the right side. For example, $s_{k,i,j}^{\alpha+} \leftarrow (s_{k,i}^{\alpha+}, w_{k,i-1,j}^{\beta})$ denotes that $s_{k,i,j}^{\alpha+}$ is constructed from $s_{k,i}^{\alpha+}$ and $w_{k,i-1,j}^{\beta}$, and $s_{i,j}^{\alpha+} \leftarrow (s_{i,j}^{\alpha+}, W_{i-1,j}^{\beta})$ indicates that each member $s_{i,j}^{\alpha+}$ is constructed in this way. In general, if $r^* \leftarrow (r, w)$ there are the following relationships among the record members:

\[
\begin{align*}
    r \cdot s\text{len} & = w \cdot s\text{len} \quad (11a) \\
    r \cdot f\text{req} & = r \cdot f\text{req} \quad (11b) \\
    r^* \cdot s\text{len} & = r \cdot s\text{len} + w \cdot d\text{len} \quad (11c)
\end{align*}
\]

where $r$ may be any static gap record in $R, S, S^+$ or $T$. Equations (11b) and (11c) explicitly express the function of operator $\leftarrow$. Table I lists the complete set of combinations of static and dynamic gap state variables used to calculate $g_{i,j}^{\beta}$ with the four running gap states defined above. Since each induction and reconstitution process takes $O(|R|)$ operations, the gap-opening cost for a path at a node is obtained with $O(|R_i^\alpha| + |R_j^\beta|)$ steps, including the calculation of the inner products defined by equation (7).

**Modification of algorithm (D)**

There are two major differences in algorithm (D) compared to algorithm (C). First, each node retains a variable number of partial paths (candidates) that potentially contribute to the global optimal path. The superscript $\alpha$ now indices each candidate, and so it corresponds not to an edge but to an edge pair $\beta$ in algorithm (C). Second, survival of a candidate is judged by criteria more complicated than those shown in equations (10). To examine these criteria, we use three additional variables, $E_{i,j}^\alpha$, $F_{i,j}^\alpha$, and $G_{i,j}^\alpha$. Their original definitions (equations 12a–c) imply $O(M \times N)$ steps for calculation of each variable. Like $D_{i,j}^\alpha$, however, $E_{i,j}^\alpha$, $F_{i,j}^\alpha$, and $G_{i,j}^\alpha$ are obtained with $O(|R_i^\alpha| + |R_j^\beta|)$ steps with generalized profiles:

\[
\begin{align*}
    E_{i,j}^\alpha & = D_{i,j}^\alpha + v \cdot \sum_{m=1}^{M} \sum_{n=1}^{N} (Q_{m,i,j}^\alpha \leq Q_{n,i,j}^\beta) \\
    F_{i,j}^\alpha & = D_{i,j}^\alpha + v \cdot \sum_{m=1}^{M} \sum_{n=1}^{N} (Q_{m,i,j}^\alpha \geq Q_{n,i,j}^\beta) \\
    G_{i,j}^\alpha & = D_{i,j}^\alpha + v \cdot \sum_{m=1}^{M} \sum_{n=1}^{N} ((Q_{m,i,j}^\alpha \leq Q_{n,i,j}^\beta) \\
    & \quad + (Q_{m,i,j}^\alpha \geq Q_{n,i,j}^\beta)) \\
    & = E_{i,j}^\alpha + F_{i,j}^\alpha - D_{i,j}^\alpha
\end{align*}
\]

where $R_{i,j}^\alpha \leftarrow (R_i^\alpha, W_{i,j}^{\alpha})$ and $R_{i,j}^{\alpha+} \leftarrow (R_j^\beta, W_{i,j}^{\beta})$.

The criteria (Gotoh, 1993) for a candidate to be retained or not involve several logical sums such as:

\[
C_1(\alpha, \beta) = \bigvee_{m=1}^{M} \bigvee_{n=1}^{N} (Q_{m,i,j}^\alpha > Q_{n,i,j}^\beta) \cdot (Q_{m,i,j}^\alpha \leq Q_{n,i,j}^\beta)
\]

and

\[
C_1(\alpha, \beta) = \bigvee_{m=1}^{M} \bigvee_{n=1}^{N} (p^\alpha \cdot s\text{len} > r^\alpha \cdot s\text{len})
\]

\[
\times (p^\beta \cdot s\text{len} \leq r^\beta \cdot s\text{len})
\]

Although a direct implementation of equation (13b) takes $O(|R_i^\alpha| \times |R_j^\beta|)$ operations in the worst case, we can devise an $O(|R_i^\alpha| + |R_j^\beta|)$ algorithm taking advantage of the ordered structure of sets $R$ and $R^\beta$. For each $r_{i,j}^\alpha \in R_i^\alpha$ and a candidate $\alpha$, we get the index $K^\alpha(r_{i,j})$ so that $r_{i,j}^\alpha > r_{i,j}^\alpha$ for any $k \geq K^\alpha(r_{i,j})$. If such an index does not exist, $K^\alpha(r_{i,j})$ is set to be $\infty$ (actually any number greater than $|R_j^\beta|$). Since $R_i^\alpha$ and hence $R_j^\beta$ have also been sorted, $O(|R_i^\alpha| + |R_j^\beta|)$ comparisons are sufficient to get $K^\alpha(r_{i,j})$ for all $r_{i,j}^\alpha \in R_i^\alpha$. Then, the logical sum

\[
C_1(\alpha, \beta) = \bigvee_{r \in R_i^\alpha} (K^\alpha(r) > K^\beta(r))
\]
is easily shown to be equivalent to those of equations (13a) and (13b). All other logical sums involved in the test routines are obtainable in similar procedures, while some procedures use the boundary that indexes $r_{k_{i,j}l}$ in addition to $K'(r_{k_{i,j}})$ that indexes $r_{k_{i,j}>l}$. Thus, all the routines that consumed $O(M \times N)$ steps in the earlier algorithm (D) are now performed in $O(R_f^A + |R_f^A|)$ steps.

**Implementation**

**Calculation of SP score**

Before assessing the performance of the new group-to-group alignment algorithms, I examined usefulness of the generalized profiles on calculation of the SP score of a single group. Various numbers of sequences were extracted from the cytochrome P450 protein superfamily (Nelson et al., 1993) to compose a set of test groups. When the conventional algorithm equivalent to equations (8) and (3) were used, the calculation time was quadratic as a function of the number of sequences in the group, $M$ (Figure 3, closed squares). When we obtained $\sum_{m=2}^M \sum_{n=1}^{m-1} b_{m,n,i}$ with the generalized profiles but $\sum_{m=2}^M \sum_{n=1}^{m-1} d(a_{m,i}, b_{n,i})$ in the literal way (half-profile method), the CPU time (open circles) was slightly longer than that with the conventional method for small values of $M$. However, the relative CPU times were soon reversed within the range of $10 < M < 20$ (Figure 3, inset). Since the CPU time with the profile-based methods depends not only $M$ but also $|S_i|$ and $|T_j|$ (equation 9), the exact intersecting point varies with the sequence groups examined. For $M$ above this range, the half-profile method is clearly superior to the conventional one, though the CPU time still grows quadratically with $M$. When both terms of equation (8) were obtained with the generalized profile vectors (full-profile method), the CPU time was linearly dependent on $M$ (closed triangles). An implementation based on equation (9a) rather than (9b) was used for the test. Compared with the other two methods, the full-profile method is more time consuming for small groups but is much more economical for larger groups. The reversal in the CPU time spent by the half- and full-profile methods occurred within the range $20 < M < 30$ (Figure 3, inset). The above results suggest the most economical way to obtain the SP score for a give group with $M$ protein sequences; use the conventional method when $M < M_u \approx 15$, the full-profile method when $M > M_l \approx 25$, and the half-profile method when $M_l \leq M \leq M_u$. Examinations with nucleotide sequences suggested the same value for $M_l$ and a slightly smaller value for $M_u$.

**Group-to-group sequence alignment with new algorithms**

To test the performance of the new algorithms, I implemented four calculation modes on the basis of each of algorithms (C) and (D). We will refer to these modes as the conventional mode, the half-profile mode, the 3/4-profile mode and the full-profile mode by analogy to the previous terminology. The conventional mode is equivalent to that described in the earlier paper (Gotoh, 1993). The half-profile mode uses the profile-based algorithm to obtain the gap costs, but calculates $\sum_{m=1}^M \sum_{n=1}^{N-1} d(a_{m,i}, b_{n,i})$ in the literal way. The sum is obtained by the use of equations (4a) and (4b) with the 3/4- and the full-profile modes respectively.

I compared the efficiency of these calculation modes with various combinations of the groups of protein sequences examined above. The primary parameter that affected the relative efficiency was the product, $M \times N$, of the numbers of sequences comprising the two groups to be aligned. As shown by closed squares in Figure 4(A), the CPU time linearly increased as a function of $M \times N$ when the conventional calculation mode was used in combination with Algorithm (C). When the half-profile mode was used, the rise in CPU time was retarded until $M \times N$ reached $\sim 100$. In contrast to the above two modes, the CPU time was only weakly dependent on $M \times N$ when the 3/4- or the full-profile modes were employed (for the sake of clarity, the results obtained with the 3/4-profile mode are not shown in Figure 4). Thus, the general tendency of the relative efficiency is similar to that observed in the calculation of the SP score of a single group. Namely, as $M \times N$ gets larger, the most economical method varies in the order of the
conventional mode, the half-profile mode, the 3/4-profile mode and then the full-profile mode. The actual intersecting points observed by the experiments shown in Figure 4(A) for the conventional versus half-profile, half-profile versus 3/4-profile, and 3/4-profile versus full-profile modes were $M \times N \approx 6, 25$ and $2000-3000$ respectively. Similar examinations with nucleotide sequences indicated that the corresponding intersecting points are $M \times N \approx 6, 20$ and $50-1000$ respectively. It should be noticed that these values can vary with the contents and distribution of internal gaps in each group. Although we may estimate more accurate intersecting points by taking the average values for $|S^A|$, $|T^A|$, $|S^B|$ and $|T^B|$ into consideration, the reward of this elaboration seems to be insignificant.

The original algorithm (D) requires considerable memory to accommodate variable numbers of candidates. Since each candidate $\alpha$ holds the sets of gap-state variables, $Q_{m_{i,j}}^A$ ($1 \leq m \leq M$) and $Q_{n_{i,j}}^B$ ($1 \leq n \leq N$), the space requirement is $O((M + N) \times \max(i, j) \times \text{maximum number of candidates at a node})$. This space exceeded the capacity of our machine for some pairs of groups in the same test set as examined above (Figure 4B). With the new algorithms, the number of candidates itself is unchanged, but the space requirement is much reduced because each candidate holds only the minimum subset of the dynamic gap states, $W_{i,j}^C$, whose average size is $< 2.0$ in typical cases. As a consequence, we could obtain the optimal group-to-group alignment for all the pairs examined with algorithm (C). The scores obtained with algorithm (D) were better than those with algorithm (C) for $\sim 30\%$ of group pairs examined.

Since the calculation time of algorithm (D) is affected by complicated factors, the plots of CPU time against $M \times N$ (Figure 4B) were more scattered than the corresponding plots observed with algorithm (C) (Figure 4A). Nevertheless, the general tendency was preserved, and the intersecting points of preferable calculation modes were nearly the same as those with algorithm (C). Algorithm (D) was $1.5-2$ times slower than algorithm (C) with any calculation modes.

**Multiple group alignment by randomized iterative strategy**

The randomized iterative strategy was proposed by Berger and Munson (1991) to align several sequences simultaneously. At each iteration step, a prealigned set of sequences is divided into two groups in a random fashion, and then the groups are realigned by a pairwise method. If we use algorithm (D) for the realignment, the overall SP score is improved monotonously and converges to a certain value (Gotoh, 1993). The same strategy is applicable to simultaneous alignment of several groups. Since the former obstacle as to computation time has now been removed, we can test the performance of this strategy. For this purpose, I used again the cytochrome P450 protein superfamily, and collected all the mammalian microsomal P450 sequences listed by Nelson et al. (1993) except CYP5 (thromboxane synthase). CYP5 was omitted because only one sequence belonging to this family is currently known. The total of 119 sequences was classified into eight families, where members of different families share 15-30\% identical amino acids (Nelson et al., 1993). Figure 5 shows the course of refinement with five independent iteration series. As expected, the SP score was monotonously improved or remained unchanged at each iteration step. The converged SP scores and alignments of the independent series were very similar to each other, implying good reproducibility of the strategy. About an hour of CPU time of a Sun workstation was taken for convergence of each series of iteration. The average value for $M \times N$ of the total of 1963 pairwise alignments in the five series was 2362. As judged from the rate of acceleration (Figure 4), we can estimate that the new algorithm enabled nearly 100-fold faster calculation in this example.

**Discussion**

The major advancement of the group-to-group sequence alignment algorithms described in this paper from the earlier versions (Gotoh, 1993) is reduction in computer time without a loss of rigoroussness. I have proposed four calculation modes — on the base of each of algorithms (C) and (D) (Gotoh, 1993). The relative efficiencies of these modes are primarily dependent on the product of the numbers of sequences comprising the two groups. Hence, we can choose the most economical (or equivalently efficient) calculation mode for a given pair of groups. The benefit of using the new algorithms becomes more prominent as the group sizes grow larger. When each of the two groups is composed of 50 sequences, the new...
algorithm runs ~100 times faster than the earlier one (Figure 4). Thus, we can solve a larger scale of alignment problems without investment in expensive hardware. The programs (alp/alm and rrp/rrn) are available from the author upon request.

As for algorithm (D), reduction in memory requirement is also significant. A candidate of the new algorithm holds only the dynamic part rather than the complete set of gap state variables. Moreover, the preprocessing to derive the static gap states provides a better estimate of the maximum number of candidates generated in the induction process. As a consequence of the combined effects, at least an order of magnitude less memory is now required to keep the complete gap state information in cases where the conventional method failed due to the memory problem.

Multiple group alignment as exemplified in Figure 5 may be the most promising application of the group-to-group alignment algorithms, especially when the groups are distantly related to one another. Simple string matching is not fully reliable in such cases, since it is known that the accuracy of alignment of a pair of protein sequences drops sharply as the amino acid identities fall below 25% (Sander and Schneider, 1991). If the three-dimensional structure of one (or more) of the sequences is known, the accuracy of alignment would be improved by incorporating physico-chemical compatibility with the template structure (Bowie et al., 1991; Godzik and Skolnick, 1992; Jones et al., 1992). Another strategy is to align simultaneously as many homologous sequences as possible. This strategy makes sense because homologous proteins (or structural RNAs) almost certainly share a common folding pattern, and the larger information contents considered at a time would reduce statistical noise. Even though the full-blown dynamic programming method is too time consuming in most practical cases (Altschul, 1989), similar effects are gained by the multiple group strategy. Inclusion of the structure-based information in the multiple group alignment appears to be the most important next step in the field of alignment of biological sequences.

Acknowledgements

This work was partly supported by a Grant-in-Aid for Scientific Research on Priority Areas, ‘Genome Informatics’, from the Ministry of Education, Science and Culture of Japan.

References

Alignment of groups of sequences


Received on July 27, 1993, accepted on February 23, 1994.