Adv. Biophys., Vol. 36, pp. 159–206 (1999)

MULTIPLE SEQUENCE ALIGNMENT: ALGORITHMS AND APPLICATIONS

OSAMU GOTOH

Saitama Cancer Center Research Institute, Ina-machi, Saitama 362-0806, Japan

A central theme of modern molecular biology is to elucidate the interrelationships among genetic information, higher-order structures of gene products, and their biological functions. Because of the inheritable nature of the genetic information, all of these properties are under the influence of molecular evolution. For short, we call the relationships among function, evolution, sequence, and structure "FESS" relationships.

Studies on the FESS relationships are becoming more and more important, yet they are complicated and demanding with explosive accumulation of structural and functional data for genes and proteins of various organisms. Most symbolically, the sizes of DNA and protein sequence databases are exponentially increasing, especially with the steady progress of genome projects. The three-dimensional (3D) structural databases are also growing rapidly, chasing the sequence databases. It is absolutely necessary to develop computational tools for storing and retrieving these massive amounts of data. However, an even more important and challenging research theme is to analyze the data and extract the essence vaguely encoded in the mass of information.

Unquestionably, it is of primary importance to decipher genomic DNA or protein amino acid sequences in relation to their

structures and functions, since all genetic information is written as DNA sequences in all living organisms, and the potential for a protein to fold into a specific tertiary structure is encoded in its primary sequence. Although "direct" approaches such as energy minimization for structural prediction obviously possess their own vital parts, "comparative" approaches attract increasing attention with wide recognition that evolutionary constraints strongly operate in the FESS relationships at the molecular level.

The genes or gene products that are derived from a common ancestor are called "homologous" to one another. Billions of genes or proteins in all the living organisms on the earth could be classified into these homology groups, often called families or superfamilies. Multiple sequence alignment (or simply multiple alignment) has been known as one of the most useful tools in study of the FESS relationships for members within a family, because it compactly represents conserved or variable features among the family members. Although sequences can extensively diversify during the long evolutionary processes, the 3D structures of proteins are usually much more conservative than sequences. Two expectations then follow. First, the 3D structure of any member of the family could be predicted through an alignment between the target sequence and one or more members of the known 3D structure. Consideration of many homologous sequences at a time, rather than just two, generally improves the reliability of their alignment. Second, unknown principles or rules of protein folding might be uncovered by analyzing the homologous sequences, since their capacity to fold into similar tertiary structures must be commonly encoded in the divergent sequences. Leaving the structural aspects aside, locally conserved regions in a multiple alignment are very likely to have some biologically important functional roles, as "the neutral theory of genes" predicts (*1*).

However, it is far from trivial to obtain a reliable multiple alignment from a set of remotely related homologous sequences. To develop automatic computational methods, multiple alignment is formulated as a combinatorial optimization problem. Although numerous methods based on various principles have been proposed, we are still seeking better ones in terms of accuracy and computational speed. This review is primarily devoted to a summary of various endeavors to this task. In a later section, the roles of mul-

tiple alignment in studies of FESS relationships will be discussed with a few examples. Since the related areas are already enormous, the topics covered in this review must be severely limited. Algorithmic aspects of global alignment of protein sequences are the major subjects discussed here. Many books and reviews are now available for readers who are interested in wider and deeper perspectives in the rapidly expanding fields (*2–9*).

I. THE PROBLEM

1. Formalization

Consider N homologous sequences $\bar{\mathbf{s}}_1 = s_{11}s_{12} \ldots s_{1L_1}$, $\bar{\mathbf{s}}_2 = s_{21}s_{22} \ldots s_{2L_2}$, ..., $\bar{\mathbf{s}}_N = s_{N1}s_{N2} \ldots s_{NL_N}$, where s_{nl} ($n \in [1, N]$, $l \in [1, L_n]$) denotes a single nucleotide or amino acid (collectively called a "residue"), and an upper bar indicates a row vector. The set of valid residue types is denoted by Σ, *e.g.* $\Sigma = \{A, C, G, T\}$ for DNA sequences. During the evolutionary processes, some residues may stay unchanged while some may change to other types. For simplic-

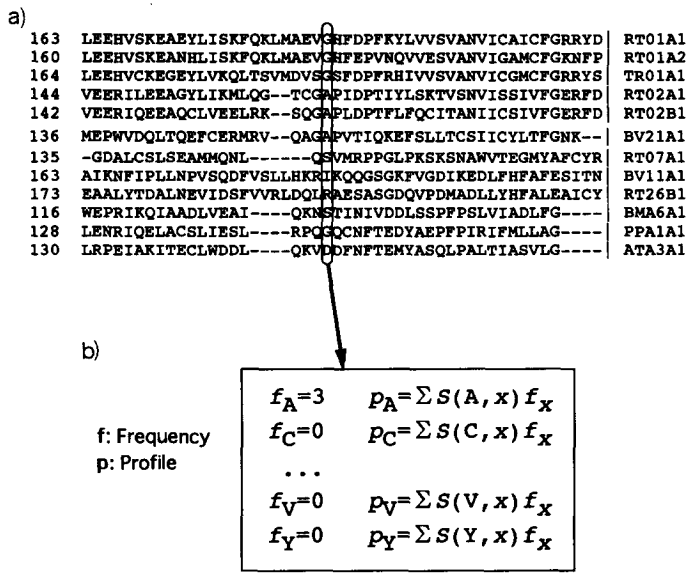

Fig. 1. An example of multiple sequence alignment (a) and frequency and profile vectors (b) derived therefrom.

ity, we regard conservation as substitution to itself. In addition, some residues may be lost, or new residues may insert somewhere between or outside the preexisting sites. The final result of the series of mutational events (substitutions, insertions, and deletions) is conveniently represented by a two-dimensional character matrix, an "alignment" $\mathcal{A} \doteq \{a_{n,i}\}$ ($n \in [1, N]$, $i \in [1, I]$), in which $a_{n,i} = s_{n,l}$ or $- \in \Sigma^* \equiv \Sigma \cup \{-\}$ is arranged so that the residues derived from a common ancestor are juxtaposed in the same column (Fig. 1a). When an insertion or deletion has produced missing residues, the corresponding elements of the matrix are padded by null characters denoted by dashes (–). Thus, each row turns out to be a contemporary sequence if these null characters are removed. Usually an insertion and a deletion are not (or cannot be) distinguished. Moreover, a single insertion/deletion event may involve several residues at a time. Hence a run of nulls is often regarded as a single entity referred to as a gap or an "indel" (abbreviation of insertion and deletion (*2*)). In this article, the terms of gap and indel are used almost interchangeably but clearly distinguished from a null indicating absence of a single residue.

If we knew all the evolutionary processes, we could unambiguously summarize them into an alignment. However, we usually know only the "results" of the processes. Our purpose is therefore to obtain the alignment that most probably represents this unknown "correct" alignment. Similarity in some biological features of independent evolutionary origins is called "analogy" and distinguished from "homology" as defined above. Significant similarities in nucleotide or amino acid sequence level come almost certainly from evolutionary relatedness, so we have formulated our purpose to recover the homologous relationships. However, the methods introduced below may be applicable to comparison of non-homologous sequences, if one keeps the limited justification in mind.

2. *Scoring Systems*

To evaluate relative correctness of an alignment, we assign a "goodness score" to each alignment, and look for the "optimal alignment" that has the best score among all possible alternatives. The score must properly reflect the expected correctness, and also must be easily calculated. In fact, the scoring system determinately affects the design of optimization algorithms. Hence, our first and very

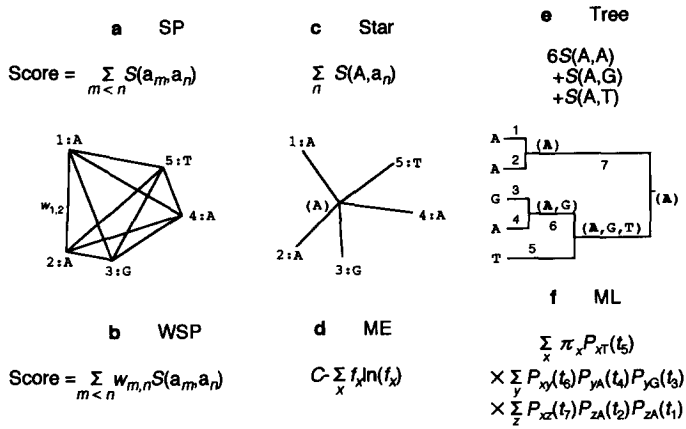

Fig. 2. Six systems for scoring a multiple alignment. (a) Sum-of-pairs (SP) score. (b) Weighted sum-of-pairs (WSP) score. (c) Star or consensus score. In this example, the most common residue 'A' is placed at the center of the star-like tree. (d) Minimum entropy (ME), where C is the baseline entropy calculated from the average abundance of residues. (e) Tree alignment score. The residue(s) deduced to reside at each internal node is represented in parentheses. The residues in bold face indicate those involved in the most parsimonious evolutionary changes. (f) Maximum likelihood score. π_x denotes the prior distribution of residue type x, and $P_{xy}(t_b)$ indicates the transition probability from residue type x to y during the period of t_b. The summations are taken over all members in (a–c), whereas over all residue types in (d) and (f).

important task is to develop a good scoring system. Several scoring systems have been proposed as summarized in Fig. 2. One group (star or consensus, sum-of-pairs (SP), and minimum entropy (ME) scores) does not, whereas the other group (tree, weighted sum-of-pairs (WSP), and maximum likelihood (ML) scores) assumes given phylogenetic relationships among the sequences to be aligned. In this article, we assume that the SP (or WSP, which is calculated in nearly the same way as SP) score is used as our target of optimization unless otherwise specified. We have chosen these scoring systems since they conform to accelerated computation, reasonably reflect correct alignments, and easily incorporate proper gap penalties.

3. *Substitution Matrix*

We have assumed that a substitution or an indel constitutes an

elementary step of an evolutionary process. In the ML framework, the probability of occurrence of each step is explicitly computed, but this is computationally very expensive. In other scoring systems, a prespecified cost is assigned to each substitution and indel. A matrix of costs of substitution for every pair of residues $\in \Sigma^*$ is called a substitution matrix. Figure 3 shows an example of 16×16 substitution matrix, S_{ab}, of nucleotides including ambiguous codes. Note that the matrix reflects the well-known observations that transitions (substitutions between purine and purine or pyrimidine and pyrimidine) are more frequent than transversions (substitutions between purine and pyrimidine), in addition to the inclusion set relationships such as R = {A, G}.

Rational derivation of an amino-acid substitution matrix stems from the work of Dayhoff and her associates (*10*). They collected amino acid sequences of closely related proteins, and counted the number of coincidence of every pair of residues observed at the homologous sites. Because of the closeness in relationship, the alignments were unambiguous and each amino-acid change could be regarded as due to a single evolutionary event. Considering the general abundance and intrinsic relative mutability of each amino acid, they converted the raw frequency data into a stationary and reversible Markovian transition matrix, $M(a, b)$. The symmetric

		A	C	M	G	R	S	V	T	W	Y	H	K	D	B	N	-
A	A	2	-2	0	-1	1	-2	-1	-2	0	-2	-1	-2	-1	-2	-1	-2
C	C	-2	2	0	-2	-2	0	-1	-1	-2	1	-1	-2	-2	-1	-1	-2
M	AC	0	0	0	-2	-1	-1	-1	-2	-1	-1	-1	-2	-2	-2	-1	-2
G	G	-1	-2	-2	2	1	0	-1	-2	-2	-2	-2	0	-1	-1	-1	-2
R	AG	1	-2	-1	1	0	-1	-1	-2	-1	-2	-2	-1	-1	-2	-1	-2
S	CG	-2	0	-1	0	-1	0	-1	-2	-2	-1	-2	-1	-2	-1	-1	-2
V	ACG	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-1	-2	-1	-1	-1	-2
T	T	-2	-1	-2	-2	-2	-2	-2	2	0	1	-1	0	-1	-1	-1	-2
W	AT	0	-2	-1	-2	-1	-2	-2	0	0	-1	-1	-1	-1	-2	-1	-2
Y	CT	-2	1	-1	-2	-2	-1	-2	1	-1	0	-1	-1	-2	-1	-1	-2
H	ACT	-1	-1	-1	-2	-2	-2	-1	-1	-1	-1	-1	-2	-1	-1	-1	-2
K	GT	-2	-2	-2	0	-1	-1	-2	0	-1	-1	-2	0	-1	-1	-1	-2
D	AGT	-1	-2	-2	-1	-1	-2	-1	-1	-1	-2	-1	-1	-1	-1	-1	-2
B	CGT	-2	-1	-2	-1	-2	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-2
N	ACGT	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2
-	-	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	-2	0

Fig. 3. A substitution matrix of nucleotides. Ambiguous residues are indicated in the standard one-letter codes.

PAM_N (Accepted Point Mutations after N substitutions per 100 residues, on average) matrix is then obtained by

$$PAM\_N(a, b) = \log_{10} \{M^N(a, b)/M^\infty(a, b)\} \qquad (1)$$

Thus, $PAM\_N(a, b)$ is the log-likelihood ratio of the expected frequency of the residue pair a and b in sequences after the average number of substitutions of N per 100 sites relative to the random distribution of these residue types. Note that $M^\infty(a, b)$ converges to the general relative abundance of residue a. A large variety of amino acid substitution matrices have been reported so far (reviewed in ref. *11*). All the matrices widely used now were obtained after Dayhoff's method with differing datasets. Figure 4 shows the original PAM_250 matrix of Dayhoff (*10*) together with that of Gonnet *et al.* (*12*) which is known to be most useful to reproduce accurate protein sequence alignments (*13, 14*). The principle of Dayhoff's method is applied to various other fields, such as derivation of a "threading potential" used for alignment of a protein 3D structure and a sequence, and a "coding potential" used for prediction of protein-coding regions in a bulk of genomic DNA sequence (Section VI-4).

		A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	-	
A	18		24	-6	-3	-3	5	-2	0	5	-8	-8	-12	-4	-6	-22	3	11	6	-35	-22	-20	A
R	-15	60		46	3	-3	-22	15	4	-10	6	-24	-22	27	-17	-32	-8	-2	-2	-16	-17	-20	R
N	1	0	20		37	22	-17	6	8	4	12	-27	-30	8	-22	-30	-8	8	5	-35	-13	-20	N
D	2	-12	20	38		46	-32	8	27	1	4	-37	-40	5	-30	-45	-6	5	0	-51	-27	-20	D
C	-20	-36	-36	-51	11		11	-24	-30	-20	-12	-11	-15	-27	-8	-8	-30	1	-5	-10	-5	-20	C
Q	-4	12	7	16	-53	40		27	17	-10	12	-18	-16	15	-10	-25	-2	2	0	-27	-17	-20	Q
E	3	-10	14	34	-52	24	38		35	-8	4	-27	-27	12	-20	-39	-5	2	-1	-43	-27	-20	E
G	12	-25	3	5	-33	-12	1	47		65	-13	-45	-44	-11	-35	-51	-16	4	-11	-40	-40	-20	G
H	-13	15	15	6	-34	29	6	-21	65		60	-22	-18	6	-12	-1	-11	-2	-3	-8	22	-20	H
I	-5	-20	-18	-23	-23	-20	-20	-25	-24	45		40	27	-20	25	10	-25	-17	-6	-17	-6	-20	I
L	-18	-29	-28	-39	-60	-17	-33	-40	-20	24	59		40	-20	27	20	-22	-20	-12	-6	0	-20	L
K	-11	33	9	0	-54	7	0	-16	0	-19	-28	46		32	-13	-32	-6	1	1	-35	-20	-20	K
M	-11	-4	-17	-26	-52	-9	-21	-27	-21	21	36	4	64		43	16	-24	-13	-6	-10	-2	-20	M
F	-35	-44	-34	-56	-43	-46	-54	-47	-18	10	18	-52	1	90		70	-37	-27	-22	35	50	-20	F
P	11	-1	-4	-9	-27	2	-5	-4	-2	-20	-25	-11	-20	-45	58		75	4	1	-50	-30	-20	P
S	11	-3	6	2	0	-5	0	10	-8	-14	-28	-1	-15	-31	9	15		22	15	-32	-18	-20	S
T	11	-8	4	-1	-22	-7	-3	0	-13	0	-16	0	-5	-31	3	13	25		25	-35	-18	-20	T
W	-57	21	-40	-67	-77	-47	-69	-69	-26	-51	-18	-34	-42	3	-55	-24	-51	17		14	40	-20	W
Y	-34	-42	-20	-43	3	-40	-42	-52	0	-9	-8	-44	-24	69	-49	-28	-27	-1	10		78	-20	Y
-	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	-20	0		0	-
	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	-		

Fig. 4. Amino-acid substitution matrices reported by Dayhoff (*10*) (lower left) and by Gonnet *et al.* (*12*) (upper right). The values shown are $10 \times PAM\_250$.

4. Gap Penalty

Treatment of indels is more compounded than that of substitutions since an unspecified number of residues may be involved in a single event. The simplest way is to disregard the continuity in inserted/deleted residues, or equivalently to assume that each single-residue indel occurs independently. Then, an indel cost can be included as an element of a substitution matrix such that $S(a, -) = S(-, a) \leqq 0$ ($a \in \Sigma$), and $S(-, -) = 0$. Alignments obtained under this simple scoring system tend to have many isolated single-residue gaps and are unlikely to realize the actual evolutionary processes. In fact, analyzing structurally aligned protein sequences, Pascarella and Argos (*15*) found that the frequencies of occurrence of longer indels rapidly decline as a function of their length k. If we assume, at the first approximation, that the distribution is geometric and indels have been generated by a single mutational event, the log likelihood ratio of observing a length-k indel relative to not observing any indel would be:

$$g(k) = \log \left\{ \frac{c \cdot \exp(-uk)}{1 - \sum_{i=1}^{\infty} c \cdot \exp(-ui)} \right\} = -(v + uk) \qquad (2)$$

where $v \equiv \log \{1/c - 1/(e^u - 1)\}$ is a positive constant. A cost for an indel of the form of Eq. 2 is often called an "affine" gap penalty. The constant term v and the proportional coefficient u are known as a "gap-opening penalty" and "gap-extension penalty", respectively. On the other hand, a cost lacking the constant term v, *i.e.* $g(k) = -u \cdot k$, is called a "proportional" gap penalty. The proportional coefficient u in either penalty function may be slightly more generalized and absorbed into the substitution matrix, as noted above, so that different types of residues should make variable contributions to a gap cost (*16*).

The affine gap costs are most widely used in current applications, because they are effective to generate reasonable alignments with minimal complexity. For long indels, however, the behavior of an affine gap function is nearly identical to that of a proportional cost. It would be natural to assume practically no difference in preference of indels of lengths 99, 100, and 101, for example. "Concave" gap penalty functions like $g(k) = -\{v + u \cdot \log(k)\}$ ($v > 0$, $u > 0$, $k \geqq 1$) have been proposed by Benner *et al.* (*17*) on the basis of

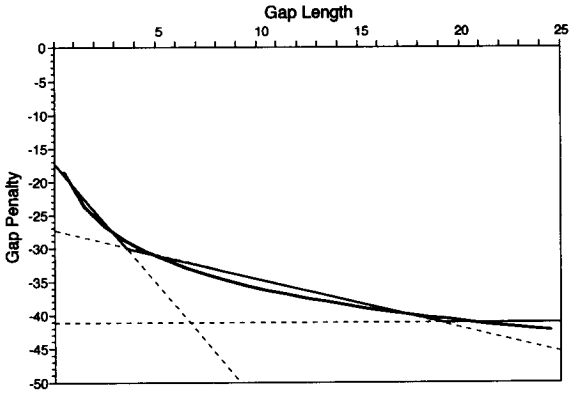

Fig. 5. Gap penalty $g(k)$ as a function of a gap length, k. The bold curve is calculated according to the equation $g(k) = -18.5 - 17.0 \cdot \log_{10}(k)$ (*17*). Solid lines show a piecewise linear gap penalty function. Note that customarily used functions significantly deviate from that shown here in both scale and shape.

distributions of gap-lengths observed in a large set of pairwise sequence alignments. Nearly the same effects are expected by a piecewise-linear function (Fig. 5).

The functional forms of an indel cost profoundly affect the efficiency of alignment algorithms. We now start with pairwise sequence alignment algorithms, which are basal for nearly all multiple alignment methods discussed later.

II. PAIRWISE SEQUENCE ALIGNMENT ALGORITHMS

1. *Alignment of Two Sequences*

One of the most basic problems in computational molecular biology is to compare two related protein or nucleotide sequences. Although the problem has been extensively studied and comprehensively reviewed (*2, 3, 8*), a brief introduction is presented here.

Consider that we are aligning two homologous protein or nucleic acid sequences $\bar{\mathbf{a}} = a_1 a_2 a_3 \ldots a_I$ and $\bar{\mathbf{b}} = b_1 b_2 b_3 \ldots b_J$, where a_i ($i \in [1, I]$) or b_j ($j \in [1, J]$) indicates a residue. We also assume $I \geqq J$ for convenience. As noted in the previous section, our purpose is to reproduce the most likely results of evolutionary changes that have occurred since the separation of $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$ from their common ancestor. Let's consider an alignment shown in Fig. 6b, in

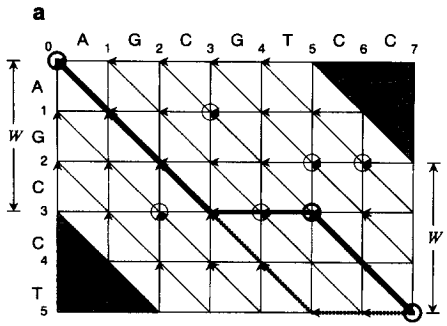

Fig. 6. Alignment path graph (a) and an optimal alignment (b). The alignment shown in (b) corresponds to the path indicated by the thick lines in (a). The other optimal path is indicated by broken thick lines. Arrows indicate the directions of traceback. An encircled node indicates the "upper left corner" of a stretch of diagonal edges. The calculation was performed with the substitution matrix shown in Fig. 3, and $g(k) = -(2k + 4)$. Off-diagonal areas that are omitted from the calculation by the "cutting corners approximation" are indicated by shading.

which we see 5 matched (conserved or substituted) sites and an indel of length 2. On the assumption of independence among sites, the log likelihood ratio of observing this alignment is

$$H = S(\mathrm{A}, \mathrm{A}) + S(\mathrm{G}, \mathrm{G}) + 2S(\mathrm{C}, \mathrm{C}) + S(\mathrm{T}, \mathrm{C}) + g(2) \quad (3)$$

where $S(a, b) = PAM\_N(a, b)$ and the background probability comes from 5 randomly matched pairs with no indel. There are many other ways to align $\bar{\mathbf{a}}$ and $\bar{\mathbf{b}}$, and the number of possible alignments amounts to $O({}_{I+J}C_I)$ (*18*). Although this number becomes huge for sequences of moderate lengths, we can efficiently find the alignment(s) that gives the largest value for the associated log likelihood ratio, usually called "alignment score", by means of dynamic programming algorithms.

The dynamic programming algorithm was first introduced by Needleman and Wunsch (*19*) in analysis of biological sequences.

The general form allowing for an arbitrary gap-penalty function is written by a recurrence equation:

$$H_{i,j} = \underset{\alpha=1,3}{\text{Max}} \{H^{\alpha}_{i,j}\} \tag{4A}$$

where

$$H^{1}_{i,j} = H_{i-1,j-1} + S(a_i, b_j) \tag{4B}$$

$$H^{2}_{i,j} = \underset{k=1,i}{\text{Max}} \{H_{i-k,j} + g(k)\} \tag{4C}$$

and

$$H^{3}_{i,j} = \underset{k=1,j}{\text{Max}} \{H_{i,j-k} + g(k)\} \tag{4D}$$

The recurrent relation is initialized by $H_{i,0} = H^{2}_{i,0} = g(i)$, $H_{0,j} = H^{3}_{0,j} = g(j)$, and $H^{2}_{0,j} = H^{3}_{i,0} = -\infty$ for $i \in [1, I]$ and $j \in [1, J]$, and $H_{0,0} = H^{2}_{0,0} = H^{3}_{0,0} = 0$. The end of the recurrence yields the alignment score $H(\bar{\mathbf{a}}, \bar{\mathbf{b}}) = H_{I,J}$. The alignment thus obtained is "global" in the sense that terminal gaps are treated equally to internal gaps. By changing the initial conditions, $H_{i,0} = 0$ for $i \in [0, I]$, and defining, $H(\bar{\mathbf{a}}, \bar{\mathbf{b}}) = \underset{i=1,I}{\text{Max}} H_{i,J}$, we obtain a "semi-global" alignment(s) (*20*), in which terminal deletions in $\bar{\mathbf{b}}$ are disregarded. Semi-global alignment is preferable to global alignment when the region homologous to sequence $\bar{\mathbf{b}}$ is totally embedded in the longer sequence $\bar{\mathbf{a}}$. While other settings of boundary conditions are possible, only global alignment problems are considered in this review, since the algorithms are nearly the same.

Although Waterman *et al.* (*21*) originally formulated the minimization of the distance metric D rather than maximization of the similarity H, their formula is equivalent to Eq. 4 (*22*). In fact, it is easily verified that Eq. 4 is invariant to the following linear transformations:

$$\begin{aligned} H'_{i,j} &= \alpha H_{i,j} + \frac{i+j}{2}\beta \\ S'(a_i, b_j) &= \alpha S(a_i, b_j) + \beta \\ g'(k) &= \alpha g(k) + \frac{k}{2}\beta \end{aligned} \tag{5}$$

where α and β are constants. We obtain the formula of Waterman *et al.* with $D_{i,j} \equiv H'_{i,j}$, $\alpha = -1$ and $\beta = \underset{a,b \in \Sigma}{\text{Max}} S(a, b)$, but as $\alpha < 0$, the Max operators must be replaced by Min operators.

The algorithm with arbitrary gap-penalty functions requires $O(I^2J+IJ^2)$, or $O(I^3)$ if $I=J$, computational steps, as the original Needleman and Wunsch method did. It is rather straightforward to derive an $O(IJ)$ algorithm with a proportional gap penalty function (*23–25*). Through a small trick, we can also devise an $O(IJ)$ algorithm with an affine gap penalty function (*26*):

$$\begin{aligned} H^2_{i,j} &= \text{Max}\,[H_{i-1,j}+g(1), \underset{k=2,i}{\text{Max}}\,\{H_{i-k,j}+g(k)\}] \\ &= \text{Max}\,[H_{i-1,j}+g(1), \underset{k=1,i-1}{\text{Max}}\,\{H_{i-1-k,j}+g(k+1)\}] \qquad (6A) \\ &= \text{Max}\,[H_{i-1,j}-v-u, \underset{k=1,i-1}{\text{Max}}\,\{H_{i-1-k,j}+g(k)-u\}] \\ &= \text{Max}\,(H_{i-1,j}-v, H^2_{i-1,j})-u \\ H^3_{i,j} &= \text{Max}\,(H_{i,j-1}-v, H^3_{i,j-1})-u \qquad (6B) \end{aligned}$$

Almost all current implementations of pairwise sequence alignment adopt the affine gap-penalty functions mentioned above. However, a little modification is necessary to generalize the affine functions to piecewise-linear functions, at the expense of reduced execution rate by only a small constant factor (*26, 27*). In brief, when $g(k) \equiv \underset{r=1,R}{\text{Max}}\,\{-(u_r k+v_r)\}$ $(u_r > u_{r+1} \geqq 0$, and $0 \leqq v_r < v_{r+1}$ for $r \in [1, R]$), Eq. 4 can be modified as:

$$H_{i,j} = \underset{\alpha=1,2R+1}{\text{Max}}\,\{H^\alpha_{i,j}\} \qquad (7A)$$

where

$$H^{2r}_{i,j} = \text{Max}\,\{H_{i-1,j}-v_r, H^{2r}_{i-1,j}\}-u_r \qquad (7B)$$

and

$$H^{2r+1}_{i,j} = \text{Max}\,\{H_{i,j-1}-v_r, H^{2r+1}_{i,j-1}\}-u_r \qquad (7C)$$

In the extreme, the penalty function may reach a constant for indels longer than some fixed length, K_{R-1} (Fig. 5). Even the simplest of such functions, $g(k)=-(uk+v)$ for $k \leqq K_1$ and $g(k)=-(uK_1+v)=$ const. for $k > K_1$ (*28*), is useful when one of the sequences contains long insertions such as introns, transposons, or interspersed repetitive elements. Although a sophisticated algorithm based on a "candidate list paradigm" which runs in $O(IJ \log (I))$ steps with more general concave gap-penalty functions has been proposed (*29, 30*), most biologically relevant problems can be solved by the easier algorithm with only a few pieces of linear function, $R \leqq 3$ (*31*).

2. Traceback

In the preceding subsection, we have discussed how to calculate the optimal alignment score. To obtain the actual alignment(s), we need to trace back the path(s) which gave rise to the optimal score. The optimal paths are most conveniently represented as a subgraph of the path matrix exemplified in Fig. 6a. Each node of the path matrix is represented by a Cartesian coordinate (i, j), where i or j points not to a residue itself but the inter-residue site behind it (exceptionally, 0 points to the site before the first residue). Several different methods have been proposed to find one or all of the optimal paths.

A) Score-based methods

The most straightforward way is to store all $H_{i,j}$ values and repeat the above-mentioned calculations in the reverse direction. Starting from (I, J), we look for the term(s) in the right member of Eq. 4 that gives rise to $H_{i,j}$. If $H_{i,j} = H^2_{i,j} = H_{i-k,j} + g(k)$, for example, we move to $(i-k, j)$. The procedure is repeated until we reach (0, 0). The trace of this walk gives an optimal path. With an arbitrary gap penalty function, the process takes $O(I^2 + J^2)$ steps. With an affine or piece-wise-linear function, $O(I + J)$ steps are sufficient to recover a single optimal alignment, provided that we have stored all $H^\alpha_{i,j}$ $(0 \leqq \alpha \leqq 2R + 1)$ values. If the optimal paths are degenerate, *i.e.*, more than one alignment is equally optimal, we can enumerate all of them by temporarily storing partial traceback paths in a stack, in the standard manner used in a depth first search through an acyclic directed graph (*32*).

A similar method is to calculate the "backward matrix" $B_{i,j}$ obtained in the same way as $H_{i,j}$ but from the opposite direction from (I, J) to (0, 0):

$$B_{i,j} = \operatorname*{Max}_{\alpha=1,3} \{B^\alpha_{i,j}\} \tag{8A}$$

$$B^1_{i,j} = B_{i+1,j+1} + S(a_{i+1}, b_{j+1}) \tag{8B}$$

$$B^2_{i,j} = \operatorname*{Max}_{h=1,\, n-i} \{B_{i+h,j} + g(h)\} \tag{8C}$$

and

$$B^3_{i,j} = \operatorname*{Max}_{h=1,\, n-j} \{B_{i,j+h} + g(h)\} \tag{8D}$$

The initial conditions are $B_{I,J} = 0$, $B_{i,J} = B^2_{i,J} = g(I-i)$, $B_{I,j} = B^3_{I,j} = g(J-j)$, and $B^3_{i,J} = B^2_{I,j} = -\infty$ for $i \in [0, I-1]$ and $j \in [0, J-1]$. Look-

ing for the node (i, j) for which $C_{i,j} = H_{I,J}$, we can trace the optimal path, where

$$C_{i,j} \equiv H_{i,j} + B_{i,j} - g(E_{i,j}) - g(F_{i,j}) + g(E_{i,j} + F_{i,j}) \qquad (9)$$

The subtraction of $g(E_{i,j}) + g(F_{i,j}) - g(E_{i,j} + F_{i,j})$ compensates for non-proportional gap penalty values, where $E_{i,j}$ and $F_{i,j}$ are lengths of gaps that merge at (i, j) from the forward and backward directions, respectively. Although this traceback method is less efficient than that mentioned above, calculation of both forward and backward matrices leads to useful algorithms, some of which will be discussed later.

B) Bit maps

More efficient traceback is achieved by recording the "direction(s)" from which the optimal path was selected at each step of recurrence. Let a Boolean variable $E^{\alpha}_{i,j} \equiv H_{i,j} = H^{\alpha}_{i,j}$ ($\alpha \in [1, 3]$). We assume that a Boolean value takes 1 if true and 0 otherwise. In the traceback routine, we move to $(i-1, j-1)$, $(i-1, j)$ or $(i, j-1)$ depending on $E^1_{i,j} = 1$, $E^2_{i,j} = 1$, or $E^3_{i,j} = 1$, respectively. $E^{\alpha}_{i,j}$ may be compactly encoded into a bit map with three bits per node. We must be cautious, however, that in this way an optimal alignment(s) can be correctly recovered only when a proportional gap penalty function is used. With more general gap penalty functions, $E^2_{i-l,j}$ may be false for some $l \in [1, k-1]$ when $H_{i,j} = H_{i-k,j} + g(k)$, and hence the traceback route could be disrupted. Of course, similar situations can arise for the horizontal direction, *i.e.*, $E^3_{i,j-l} = 0$ for $\exists l \in [1, k-1]$ when $H_{i,j} = H_{i,j-k} + g(k)$. Gotoh (*26*) proposed an *ad hoc* method involving local traceback routines during the forward process. Although the method successfully recovered at least a single optimal alignment, it was inefficient and failed to reproduce all alternative optimal alignments if desired. Altschul and Erickson (*33*) devised a more elaborate method which uses seven Boolean variables per node, instead of just three, in the case of affine gap penalties. With the Altschul-Erickson method, all and only the optimal alignments can be obtained without extra local traceback during the forward phase.

C) Gap-state variable

Taylor (*34*) took a slightly different way. In his method, $E^1_{i,j}$ is the same as that defined above, while $E^2_{i,j}$ or $E^3_{i,j}$ is no longer Boolean but indicates the length of gaps up to the current node. $E^2_{i,j}$ follows

the simple recurrence equation:

$$E^2_{i,j} := 1 \quad \textbf{if } H^2_{i,j} = H_{i-1,j} + g(1) \qquad (10)$$
$$:= E^2_{i-1,j} + 1 \quad \textbf{otherwise}$$

The recurrence equation of $E^3_{i,j}$ is obvious by symmetry. We will call $E^2_{i,j}$ or $E^3_{i,j}$ a "gap-state variable", which plays an essential role in efficient multiple sequence alignment algorithms as discussed more extensively in the next section.

D) Linked-lists

Although Taylor's method can be more efficient than that of Altschul and Erickson, it may fail to enumerate all and only the optimal alignments. To amend this defect, Gotoh (*27*) proposed yet another method, in which traceback information is stored in the form of linked lists. A working variable used in place of a gap state is a pointer to the list that stores the coordinate of the latest "corner" of the alignment path (Fig. 6a). Note that the corners of a path possess sufficient information to regenerate the complete alignment. The traceback procedure naturally produces the complete set of optimal alignments in the form of a directed acyclic graph. The greatest advantage of this method over others resides in its versatility, being applicable to any forms of gap penalty functions with minor revisions.

E) Linear-space algorithm

The linear-space traceback algorithm was first introduced by Hirshberg (*35*) to solve the longest common subsequence problem, and later redesigned by Myers and Miller (*36*) to fit sequence alignment. While all the methods mentioned above use $O(IJ)$ space to store traceback information, this algorithm requires only $O(J)$ memory to obtain a single optimal alignment. The method is applicable to local as well as global alignment problems, and is now widely used in popular alignment programs (*37*). Let $H_{I,J}$ be the optimal score which is obtained by the "forward" routine (Eq. 4). $H_{I,J} = B_{0,0}$ must hold (*38*) since the backward matrix $B_{i,j}$ is generated in the same way as the forward matrix simply by inverting the direction. More generally, for any row $i^* \in [0, I]$, we can find a coordinate (i^*, j^*) on an optimal alignment path at which $C_{i^*,j^*} = \underset{j \in [0,J]}{\text{Max}}\, C_{i^*,j} = H_{I,J}$, where $C_{i,j}$ is defined in Eq. 9. In particular, we find such a coordinate $(i^* = I/2, j^*)$ near the middle of the rect-

angular path matrix. Now, our problem is divided into two smaller problems of finding the next "midpoints" within $\{[0, i^*] \times [0, j^*]\}$ and $\{[i^*, I] \times [j^*, J]\}$ coordinate systems. Recursively applying the above procedure to the subproblems, we finally find all coordinates which specify an optimal alignment. This algorithm requires about twice as long as the forward calculation alone. Since modern computers are equipped with increasingly more high-speed memory, several traceback routines should be considered for optimal balance between time and space in practical applications.

3. Pairwise Alignment between Groups of Sequences

In some iterative multiple sequence alignment methods, each elementary step consists of pairwise alignment between a sequence and an already aligned group, or more generally, between groups. Let the two groups $\mathcal{A} = \{a_{m,i}\}$ ($m \in [1, M]$, $i \in [1, I]$) and $\mathcal{B} = \{b_{n,j}\}$ ($n \in [1, N]$, $j \in [1, J]$). Our purpose is to search for the optimal alignment between $\mathcal{A}$ and $\mathcal{B}$ that has the maximal SP (or WSP) score, in which each column vector acts as a unit, and a gap newly inserted between such vectors spans one or more entire column(s). Although the problem constitutes the very crucial part in iterative alignment methods, relatively little care has been devoted to precise algorithms. Although for the most part we can follow the basic procedures of pairwise alignment of single sequences as discussed above, two issues arise. First, a naive implementation may require $O(MN)$ steps merely to calculate an SP score corresponding to $S(a_i, b_j)$ in Eq. 4. Since M and N may exceed 10^2 in typical applications, compression of these steps is strongly desired. Second, existence of internal gaps invokes some complications when an affine or more general gap-penalty function is used. Fortunately, both of the issues can be resolved under an affine gap cost function (*39, 40*). Since the entire story is too lengthy to reproduce, only the essence of the ideas is presented here.

A) Calculation of SP score for aligned groups

Before discussing alignment procedure, we briefly consider how to evaluate the SP score $H(\mathcal{A}, \mathcal{B})$ between two groups of sequences within a given larger alignment $\mathcal{C} = \{c_{k,l}\}$ ($k \in [1, M+N]$, $l \in [1, L]$). We will define $a_{m,l} \equiv c_{m,l}$ and $b_{n,l} \equiv c_{M+n,l}$, allowing for some column vectors, $\mathbf{a}_i$ and $\mathbf{b}_j$, to consist of null characters alone. (In an ordinary multiple alignment, columns consisting of only nulls are

prohibited.) By definition,

$$H(\mathcal{A}, \mathcal{B}) = \sum_{m=1,M} \sum_{n=1,N} H(m, n)$$
$$= \sum_{l=1,L} \{ \sum_{m=1,M} \sum_{n=1,N} S(a_{m,l}, b_{n,l}) + v \cdot G_{m,n,l} \} \qquad (11)$$

where $H(m, n)$ is the alignment score between sequences $\bar{\mathbf{a}}_m$ and $\bar{\mathbf{b}}_n$ (matching nulls are ignored), $G_{m,n,l}$ is a Boolean variable indicating opening of a new gap between sequences $\bar{\mathbf{a}}_m$ and $\bar{\mathbf{b}}_n$ at the column position l, and v is the gap-opening penalty constant. Although the literal calculation takes $O(MNL)$ operations, the algorithm shown below runs in $O((M+N+C)L)$ steps ($C \propto |\Sigma^*| = \text{constant}$).

The data structure key to the reduction in calculation time is the "generalized profile" (*40*) which is an extension of the classical "profile" introduced by Gribskov *et al.* (*41*). For the alignment $\mathcal{A}$, we will define vector $\mathbf{f}_l^{\mathrm{A}}$ by $f_{x,l}^{\mathrm{A}} = \sum_{m=1,M} \delta(a_{m,l}, x)$ ($x \in \Sigma^*$), where $\delta(x, y) \equiv (x = y)$. We will call $\mathbf{f}_l^{\mathrm{A}}$ "frequency vector", since its element, $f_{x,l}^{\mathrm{A}}$, is the frequency of occurrence of residue x on column l of $\mathcal{A}$ (Fig. 1b). We further define "profile vector" $\mathbf{p}_l^{\mathrm{A}}$ as $p_{x,l}^{\mathrm{A}} = \sum_{y \in \Sigma^*} S(x, y) f_{y,l}^{\mathrm{A}}$. Vectors $\mathbf{f}_l^{\mathrm{B}}$ and $\mathbf{p}_l^{\mathrm{B}}$ are obtained analogously from $\mathcal{B}$. Then,

$$\sum_{m=1}^{M} \sum_{n=1}^{N} S(a_{m,l}, b_{n,l}) = \sum_{n=1}^{N} p_{b_{n,l},l}^{\mathrm{A}} = \sum_{m=1}^{M} p_{a_{m,l},l}^{\mathrm{B}} \qquad (12\mathrm{A})$$
$$= \sum_{x \in \Sigma^*} p_{x,l}^{\mathrm{A}} \cdot f_{x,l}^{\mathrm{B}} = \sum_{x \in \Sigma^*} f_{x,l}^{\mathrm{A}} \cdot p_{x,l}^{\mathrm{B}} \qquad (12\mathrm{B})$$

Depending on the values for M, N and $|\Sigma^*|$, we can choose the most economical one among the five forms of the summation.

The gap-state variable introduced in Section II-2C acts the central role in efficient calculation of $G_{m,n,l}$. Define $Q_{m,l}^{\mathrm{A}}$ as the number of consecutive nulls including and immediately preceding the position l along the row m. (The symbol Q instead of E is used here to distinguish a "static" gap state defined for an internal fixed gap from a "dynamic" gap state which is variable during the alignment process.) We also define $q_{m,l}^{\mathrm{A}} \equiv (a_{m,l} = -)$. $Q_{n,l}^{\mathrm{B}}$ and $q_{n,l}^{\mathrm{B}}$ are defined analogously. As illustrated in Fig. 7, $G_{m,n,l}$ is obtained by (*39*):

$$G_{m,n,l} = (1 - q_{m,l}^{\mathrm{A}}) q_{n,l}^{\mathrm{B}} (Q_{m,l-1}^{\mathrm{A}} \geq Q_{n,l-1}^{\mathrm{B}})$$
$$+ q_{m,l}^{\mathrm{A}} (1 - q_{n,l}^{\mathrm{B}}) (Q_{m,l-1}^{\mathrm{A}} \leq Q_{n,l-1}^{\mathrm{B}}) \qquad (13)$$

The doubly looped summation of $G_{m,n,l}$ over m and n in Eq. 11 is greatly accelerated if we consider the members with the same $Q_{m,l}^{\mathrm{A}}$

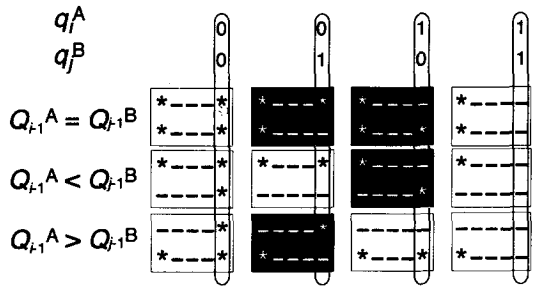

Fig. 7. Detection of opening of a gap. A reversed black and white box indicates a configuration in which a new gap opens at the enclosed sites. A dash and an asterisk denote a null and a residue other than a null, respectively. See the text for meanings of the variables q and Q.

value collectively. Just like the frequency vector $\mathbf{f}_l^A$, we will define the gap-profile $R_l^A(k)$ by $R_l^A(k) \equiv \sum_{m=1, M} \delta(Q_{m,l}^A, k)$. Figuratively speaking, this conversion is the "second quantization" of gap states. Since Q and hence k can possibly take any non-negative integer values smaller than L, $\mathbf{R}_l^A = \{R_l^A(k)\}$ should be encoded in a data structure a little more flexible than a simple array. The cardinality of this set, $|\mathbf{R}_l^A|$, equals the number of distinct $Q_{m,l}^A$ values found at the column position l, and is usually much smaller than M when M is appreciably large. Although the details are not reproduced here, we can perform the summation of $G_{m,n,l}$ in Eq. 11 in $O(|\mathbf{R}_l^A| + |\mathbf{R}_l^B|)$ (*40*).

The conversions from the original character vectors into $\mathbf{f}_l$, $\mathbf{p}_l$, and $\mathbf{R}_l$ take $O(M+N)$ operations. Equation 12B ensures that the summation is done within a constant time even when M and N are very large. Obviously $|\mathbf{R}_l^A| \leqq M$ and $|\mathbf{R}_l^B| \leqq N$, and so the summation of $G_{m,n,l}$ is done in at most $O(M+N)$ (actually in almost constant time for the majority of columns). Thus the overall calculation of $H(\mathcal{A}, \mathcal{B})$ is finished within $O((M+N+C)L)$.

B) Pairwise group-to-group alignment

Nearly the same procedure as described above is used to calculate the partial score $H_{i,j}$ in the process of group-to-group sequence alignment. Equation 12 is valid with trivial modifications to derive the SP score $S(\mathbf{a}_i, \mathbf{b}_j)$. The major difference is in the way of specifying the gap-state values which must be evaluated at each node (i, j) for each edge during the recurrence. These "working" gap states are easily obtained by combining the static gap-state variables, $\mathbf{Q}^A$

and $\mathbf{Q}^B$ or $\mathbf{R}^A$ and $\mathbf{R}^B$, and dynamic gap-state variables E^α slightly modified from those introduced in Section II-2C. Again, it is too lengthy to show here the full details, but we can calculate the precise number of gaps in $O(|\mathbf{R}_i^A| + |\mathbf{R}_j^B|)$.

The existence of internal gaps and affine gap costs assigned to individual pairs of sequences in the two groups evoke a subtle problem if we pursue exact optimal alignment between groups. In the cases of Fig. 7, for example, we do not know whether a gap exists or not until the left-to-right scan of the matching nulls reaches the end. Thus, the ordinary dynamic programming scheme, which depends only on "past" information, can fail to find the globally optimal solution. Gotoh (*39*) devised an algorithm based on the candidate-list paradigm, which had been used in the sequence alignment problem with a concave gap-penalty function (see Section II-1). The algorithm well conforms to operations with the generalized profiles (*40*). Each candidate corresponds to a partial path that arrives at a node (i, j). Like ordinary dynamic programming algorithms, these candidates are nominated from survivors in the three preceding nodes $(i-1, j-1)$, $(i-1, j)$, and $(i, j-1)$. These candidates are mutually examined for whether the progenies of one candidate can surpass the corresponding progenies of the others. If no possibility exists, such a candidate is eliminated from the list. Thus, the number of surviving candidates can vary from node to node, while the number is fixed in a usual dynamic programming procedure.

The "pruning" phase crucially affects the overall performance of the algorithm. In general, the principle of "strike while the iron is hot" works well. In Gotoh (*39*) four criteria to examine mutual superiority of candidates were proposed. The latest implementation uses simpler criteria, which are slightly more efficient than the previous ones (Gotoh, unpublished). Possibly still better methods remain to be exploited.

III. MULTIPLE SEQUENCE ALIGNMENT ALGORITHMS

1. N-dimensional Dynamic Programming Methods

Straightforward extension of the pairwise alignment methods to N-dimensional problem is conceptually easy when proportional gap-penalty costs are used, as formulated at a very early stage by

Sankoff (*42*) and Waterman *et al.* (*21*). The guiding principle of these proposals was parsimony, in which the total number of mutational events (substitutions and single-residue indels) is minimized under the constraints of a given (star or binary) tree topology. Because the computation time is proportional to $N(2L)^N$, practical applicability of the method is confined to only the cases of $N = 3$ or 4. If an affine gap penalty is used, the situation is even worse. First, affine costs are poorly compatible with the parsimony principle. Second, the computation time bursts still more rapidly with an increase in N if the SP score is used (*43*). In fact, extension to N-dimensional SP algorithms is not trivial, and the three-way alignment methods reported in the middle of the 1980's (*44–46*) differed from each other in the details of treatment of gaps (*43*).

Although full-blown N-dimensional dynamic programming methods are impractical for most applications, three-way methods have special importance as an elementary step of iterative alignment procedures, since three branches merge at an internal node of any bifurcating unrooted trees (*47, 48*). The complication associated with "natural" affine gap costs argued by Altschul (*43*) may be mitigated by the use of gap state variables (or generalized profiles) in the framework of the candidate-list paradigm. Implementation of three-way methods along this line is quite promising.

2. *Acceleration of N-dimensional Methods*

Since pairwise alignment programs run much faster than straightforward N-dimensional methods ($N > 2$), it is natural to make use of the information obtained from the $N(N-1)/2$ combinations of pairwise alignments for construction of a multiple alignment. There are several distinct approaches for this purpose. Although they have attained partial successes, none of them has been proven to have broad applicability. Five of the approaches are briefly discussed here.

A) Space-saving algorithms

The idea of speeding up sequence alignment by saving the scan space or "computational volume" (a subgraph that contains the optimal paths) is quite old. The simplest of all is the "cutting-corners approximation" (*2*). Although the optimality is not guaranteed, this approximation is free of overhead, works very well in practice, and is easily adapted for N-dimensional problems. Other

methods are usually formulated to minimize the distance $D_{\mathbf{x}} \geqq 0$ rather than to maximize similarity $H_{\mathbf{x}}$, where $\mathbf{x} = \{x_i;\ i \in [1, N]\}$ denotes an N-dimensional coordinate. For pairwise alignment, Fickett (*49*) and Ukkonen (*50*) attempted to reduce the computational volume by avoiding calculation of $D_{\mathbf{x}}$'s that would exceed an upper bound D^{U}. D^{U} may be prespecified or dynamically augmented. Carrillo and Lipman (*51*) extended this idea to N-dimensional SP alignment. They estimated the N-dimensional computational volume as the intersection of those of $N(N-1)/2$ "projections" each of which is obtained based on the formula:

$$d^*_{x_m,x_n} \leq D^{\mathrm{U}} - \sum_{\{k<l\}'} d^*_{x_k,x_l} \leq D^{\mathrm{U}} - \sum_{\{k<l\}'} D_{x_k,x_l} \tag{14}$$

where the summation is taken over all $N(N-1)/2$ pairs less (m, n), $d^*_{x_m,x_n}$ is the score associated with the "induced" alignment projected from the N-dimensional optimal alignment onto the (m, n) plane, and D_{x_k,x_l} is the score associated with the optimal pairwise alignment between the m-th and the n-th partial sequences. The first inequality derives from the proposition that the optimal alignment score is less than or equal to D^{U}, and the second inequality derives from the fact that the induced pairwise alignments are no better than the corresponding optimal pairwise alignments. Spouge (*52, 53*) observed that Min $(D_{\mathbf{x}} + \hat{B}_{\mathbf{x}}, D^{\mathrm{U}})$ provides a better estimator of the upper bound, where $\hat{B}_{\mathbf{x}}$ is an estimated upper bound of the backward matrix $B_{\mathbf{x}}$. Spouge (*52, 53*) also proposed using A* algorithm in the multiple alignment problem. A* algorithm (*54*) has been used to solve the shortest path problem more efficiently than the classical Dijkstra's algorithm (*32*). In this algorithm, an estimator of not an upper bound but a lower bound of $B_{\mathbf{x}}$ is used, and like Ukkonen's, but unlike usual dynamic programming algorithms, new paths are searched in a depth-first manner (*55*). For multiple alignment, a lower bound is obtained by $\sum_{m<n} B_{x_m,x_n}$, where B_{x_m,x_n} is a partial backward alignment score associated to the optimal pairwise alignment between sequences $\bar{\mathbf{a}}_m$ and $\bar{\mathbf{a}}_n$. The well-known MSA program (*56, 57*) adopts both the Carrillo-Lipman restriction (Eq. 14) and the A* algorithm. Imai and his group have made several extensions to the A*-based multiple alignment algorithms (*58, 59*).

B) Segment identification methods

In this review, local alignment problems are not extensively dis-

cussed, but some global multiple alignment methods use pairwise local alignment information in an earlier phase of the strategies, mainly to locate "anchor points". An anchor point means a local multiple alignment likely to take part in the optimal global alignment. Once such an anchor point is established near the middle of the final alignment, the computational volume of an N-dimensional problem could be reduced by a factor of 2^{N-1}. The more anchor points are assignable, the less subsequent computation is required. Sobel and Martinez (*60*) reported the first such trials, in which identically common segments longer than a specified length were first found by sorting (*61*) and then combined into a global alignment by a sparse dynamic programming algorithm. Santibánez and Rohde (*62*) and Vingron and Argos (*63*) used a lookup table (*64, 65*) to rapidly search for identical segments of a fixed length. Since strict segment identity among all sequences is too severe a condition (*e.g.*, only two separate loci, Phe and His, are fully conserved among proteins in the globin family), somewhat relaxed criteria were used to obtain a practical number of anchor points.

Schuler *et al.* (*66*) collected similar segments of a fixed window size in every pair of sequences, and parsed them for mutual consistency to yield one or more anchor points. The former procedure is essentially the same as that used in noise-filtration in some dot-matrix methods (*67*). Vingron and Argos (*68*) and Vingron and Pevzner (*69*) proposed to extract conserved (or consensus) segments by multiplication of dot-matrices. Depiereux and Feytmans (*70*) and Morgenstern *et al.* (*71*) also developed segment-based multiple alignment methods which did not necessarily produce a global alignment. The "consensus word" algorithms of Johnson and Doolittle (*72*) and Waterman (*73*) may be regarded as one extreme (a narrow band width and relaxed segment identification) of segment-based multiple alignment methods.

C) Consistency methods

The concept of consistency defined here is more specific than that used in the parsing phase of the segment-based algorithms mentioned above. For example, if a segment $[s_{1,i} \ldots s_{1,i+x}]$ matches $[s_{2,j} \ldots s_{2,j+y}]$ in a pairwise alignment $\mathcal{A}(\bar{\mathbf{s}}_1, \bar{\mathbf{s}}_2)$, $[s_{2,j} \ldots s_{2,j+y}]$ matches $[s_{3,k} \ldots s_{3,k+z}]$ in $\mathcal{A}$ $(\bar{\mathbf{s}}_2, \bar{\mathbf{s}}_3)$, and $[s_{3,k} \ldots s_{3,k+z}]$ matches $[s_{1,i} \ldots s_{1,i+x}]$ in $\mathcal{A}$ $(\bar{\mathbf{s}}_3, \bar{\mathbf{s}}_1)$, then the segments in the three sequences are mutually consistent. Such regions may be used as anchor points of the 3-way

alignment among $\bar{\mathbf{s}}_1$, $\bar{\mathbf{s}}_2$, and $\bar{\mathbf{s}}_3$. The non-negative integers x, y, z may not be equal to each other, but gaps must match gaps in register in such cases. Gotoh (*74*) proposed an efficient method to recognize such consistent regions among N sequences, when $N(N-1)/2$ pairwise alignments are given. Each input alignment is represented as a directed acyclic graph (Section II-2) so that all the degenerate optimal pairwise alignments are considered. Due to the combinatorial expansion together with rapid shrinkage of consistent regions with an increase in N, applicability of this method was confined to $N = 3$ or 4 appreciably divergent sequences. Miller (*75*) relaxed the condition of consistency when $N > 3$ (connected graph rather than complete graph) enabling a greater number of sequences for the analysis.

D) Reliably aligned regions

Several authors (*38, 76–78*) have discussed methods for locating reliably aligned regions in an optimal pairwise global alignment. The reliability of a matched pair or an indel along the alignment is measured on the extent of a drop in alignment score when that pair or indel is omitted during the alignment process. Intersections of "robust" regions thus found in various pairwise alignments may serve as the anchor points for multiple alignment.

Another measure of robustness is a high probability for a particular pair to match or for an indel to occur. The forward and backward "probabilistic" alignments (*79–83*) can be used to compute such probabilities (*9*).

Locating reliably aligned regions by either method is considerably more compounded than mere optimal alignment. Application of these methods to the multiple alignment problem has apparently not appeared in the literature.

E) Divide-and-conquer algorithms

The anchor points found by the methods introduced in the last three subsections generally correspond to regions conserved among all sequences to be aligned. If an anchor point happens to reside near the middle of the final alignment, the greatest speeding up is possible. If we can find a cut by which the resultant global optimal alignment is divided into two parts, the original problem is reduced to two smaller subproblems. By recursively applying the procedure to the smaller problems, we can finally track down the original problem into very small ones which may be solved directly. Of

course, finding the optimal cutting plane is as difficult as the original problem (recall the discussion in "linear-space traceback" in Section II-2E). Dress, Stoye and others (*84–86*) proposed a heuristic method which finds a cutting plane $\mathbf{x} = \{x_k;\ x_1 = L_1/2,\ k \in [2, N]\}$ that minimizes the "frustration",

$$F(\mathbf{x}) \equiv \sum_{1 \le m < n \le N} \{H_{L_m,L_n} - C_{x_m,x_n}\} \tag{15}$$

where C_{x_m,x_n} is obtained by Eq. 9 from pairwise optimal alignments. The performance of the algorithm seems to depend on the fortuity of the initially chosen sites being in a conserved region, around which the distribution of F would be narrow but otherwise the cutting plane is less confident.

3. *Progressive Methods*

The most natural application of the pairwise alignment method to an N-dimensional problem would be the progressive method. In this method two of the N sequences are aligned pairwise and the resulting alignment is "frozen" to generate a single entity. Then, two of the N-2 sequences or the alignment is aligned and the result is frozen. The process is repeated until all the N sequences are gathered into a single alignment. Many authors have proposed methods based on this principle (*28, 39, 87–95*), see also (*96–98*) for reviews. This "grass roots" approach generally works better than the sophisticated methods introduced in the previous section in terms of the quality of the results and especially of the computational speed, memory, and the number of sequences accepted.

The published methods vary in the specified order of pairwise alignments, the way of representing a frozen multiple alignment, and the scoring system or the objective function (reviewed in ref. *96*). The simplest ordering is to add individual sequences one-by-one to the intermediate aggregate in the successive (*e.g.*, alphabetically sorted, shorter to longer, or longer to shorter) order. However, most highly appreciated methods adopt the principle of "the closer the earlier". Usually a guide tree is constructed based on the distances calculated from pairwise alignments between all pairs of sequences. Sometimes this is the most time-consuming phase when N is large, but a large part of irrelevant calculations may be circumvented (*99, 100*). Given the tree, we choose a node at which two leaves corresponding to extant sequences bifurcate, obtain the

pairwise alignment of the sequences, and replace the two leaves by a new one corresponding to the resultant alignment. The process is repeated until we reach the root of the guide tree, where all sequences have merged into a multiple alignment.

In early days when computer memory was expensive, each multiple alignment was often represented by consensus or ambiguous codes (*92, 101*), which inevitably led to a loss of some information. Since computers are now equipped with much more memory, more lavish measures (SP, WSP, or ME) can be used as the objective functions. As discussed in Section II-3, we can save memory and improve computational efficiency by representing each intermediate multiple alignment as a profile or, more preferably, as a generalized profile. Additional information such as predicted secondary structures or propensities of gap formation (*102, 103*) may be supplemented to the originally defined profiles. When a tree alignment (Fig. 2e) is to be constructed (*87, 95*), each unit of an intermediate alignment is a set of residues $\in \Sigma^*$. In Hein's method (*95*), each intermediate is represented by a graph rather than a fixed alignment (see Section II-2A and -2D), and two graphs or a graph and a sequence are assembled by a pairwise "network" alignment algorithm (*2*), which is a dynamic programming algorithm more general than that used for sequence alignment. The network alignment can reduce the effects of artifacts that originate from a fixed order of pairwise alignments.

4. *Iterative Refinement Methods*

The major drawback of the progressive methods is the failure to correct for errors introduced in an early phase of the procedure. Early developers of the strategy recognized this flaw and proposed some iterative procedures to improve the quality of the alignment (*47, 87, 88*). These three initial attempts happen to correspond to three representative categories of refinement strategies: (i) consensus updating, (ii) randomized partitioning and realignment, and (iii) iterative tree reconstruction. These three strategies will be discussed below under restrictive headings.

A) Hidden Markov model

Waterman and Perlwitz (*88*) proposed to obtain the final multiple alignment by pairwise alignments between the progressively calculated consensus and individual input sequences. The consensus or

average state was represented in a way similar to the frequency vector (Section II-3A). Bains (*101*) alternately repeated pairwise alignment and updating of the consensus until convergence was attained, although the consensus state was represented by less informative single-letter codes rather than a profile.

Multiple alignment based on a hidden Markov model (HMM) (*104, 105*) is a relatively new mode of this line of iterative strategies. The model, which roughly corresponds to a consensus state, consists of three kinds of states, match states, insert states, and delete states, in addition to the start and the end states. Each state has its own emission probability (roughly corresponds to a frequency vector) and transition probabilities (roughly corresponds to gap open, close, and extension frequencies). Unlike other iterative methods mentioned above and below, the parameters corresponding to the substitution matrix and gap penalties are mostly learnt from the data themselves in parallel to the construction of the multiple alignment. To adjust parameters, either the standard expectation-maximization (EM) algorithm (*106*) or a gradient descent training algorithm (*105*) has been used.

Although HMM methods are founded on mathematically sound principles, several problems are reported for their practical applications (*107*). First, there is some lower limit ($N > \sim 50$) in the number of training sequences to assure robust learning. Second, like other hill-climbing approaches, HMM optimization can be easily trapped in local optima. Because the parameters are also adjusted in parallel to maximization of the score, the problem of local optima seems more severe than other approaches without parameter adjustments. Third, an HMM can be unstable in gap-rich regions. In fact, present HMM methods give up detailed alignment within such gap-rich regions, and involve occasional reassignments of match and insert states called "surgery", which increases complexity of the overall procedures. Finally, current HMMs do not explicitly consider evolutionary relationships among the member sequences. Since dependence on evolutionary relationships is the major source of the relatively high performance of tree-dependent progressive methods, any statistical model involving evolutionary processes is highly desired.

B) Iterative refinement methods

The methods categorized here try to optimize the objective score

(mostly SP score) by iteration more explicitly than the consensus methods mentioned above. The very early work of Sankoff *et al.* (*47*) was unique in two respects: first, it pursued optimal tree alignment rather than SP alignment, and second, a three-way dynamic programming method rather than a pairwise method was used as an elementary step of the iterative refinement. Barton and Sternberg (*91*) obtained an initial multiple alignment by a successive method. They then extracted a subalignment less one of the members, *e.g.* $\bar{\mathbf{s}}_1$, and realigned the subalignment with $\bar{\mathbf{s}}_1$ again. The process was repeated in rotation with a single excluded member, $\bar{\mathbf{s}}_n$ ($n \in [1, N]$), until no change in the score was observed. Subbiah and Harrison (*108*) extended this approach so that the initial alignment is divided into two groups consisting of $K \in [1, N/2]$ and $N - K$ members. This round-robin approach examines $2^{N-1} - 1$ ways of divisions in a fixed order, from smaller to larger values of K. The randomized iterative refinement (RIR) strategy of Berger and Munson (*109*) is very similar to that of Subbiah and Harrison, differing only in the order of divisions which are chosen from a series of random numbers. Another variant of the Subbiah-Harrison method is the "best-first" iterative refinement (BIR) strategy (Gotoh, unpublished; *110*), which chooses the one with the best score among all the pairwise alignments derived from the $2^{N-1} - 1$ ways of divisions at each stage, and continues the process until all divisions yield the same alignment score.

All these methods except for Sankoff *et al.* (*47*) appear to optimize SP score, although the explicit forms of their objective function were often not stated. Subbiah and Harrison (*108*) noted that sometimes their series of iterations did not converge. However, if we obtain the rigorous group-to-group alignment with the optimal pairwise SP score at each iterative step, the overall SP score must converge to a finite value, because the series is obviously bounded and monotonously non-decreasing (Fig. 8). As shown in Section II-3B, we can obtain rigorous pairwise alignment between large groups without extensive consumption of computer resources (*39, 40*). The problem of local optima intrinsic to this type of hill-climbing approaches is mitigated by comparison of the results from several independent series of random partitions (*39, 109*).

Since the number of possible divisions into two groups grows exponentially with N, the full-range RIR or BIR method soon be-

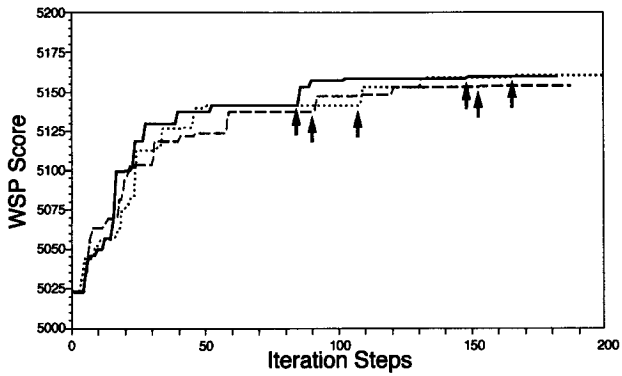

Fig. 8. Improvement in WSP score by the DNR method. Arrowheads indicate the end of an internal loop. Recalculation of the pair weights causes discontinuity in score values at these boundaries. Three independent series of randomized iterations were conducted to align 18 globin protein sequences.

comes impractical for $N > \sim 10$. Hirosawa *et al.* (*110*) examined performance of several variants of RIR and BIR strategies. They observed that the best cost-performance was obtained by "tree-based" RIR strategies, which are hybrids of the progressive method and simpler RIR strategies. After two groups of sequences totaling $K \leqq N$ members are merged at each branching node of the given guide tree, an RIR method is applied to some restricted partitioning, *e.g.* only $1 : K-1$ partitions, or "tree-restricted" partitions corresponding to the $2K-3$ edges of the subtree. Since $O(K)$ iterations are required till convergence at each node, and there are $N-1$ internal nodes in the binary guide tree, the total number of pairwise alignments amounts to $O(N^2)$. The doubly nested RIR strategy discussed below finishes with $O(N)$ pairwise alignments, while the desirable properties of the tree-based RIR strategies are inherited.

C) Doubly nested randomized iteration

The underlying principles of the progressive and tree-based RIR strategies are two fold. First, alignments between closer sequences are more reliable than those between distantly related sequences. Second, once multiple alignments of similar sequences are established, group-to-group alignment is more reliable than that between individual single members. Although these principles are built-in in the procedures of the progressive and tree-based RIR

strategies, we can incorporate the principles in the objective score functions independently of the optimization procedures. WSP score (*111, 112*) is one of the simplest of such score functions.

To obtain the pair weights, w_{mn} ($m < n \in [1, N]$), we first calculate an unrooted tree which represents evolutionary relationships among all the members of sequences to be aligned. Then, the weights are calculated to meet the following conditions, (i) closer pairs should have larger weights, (ii) a pair between "dense" members with many close relatives should be down-weighted compared to that between isolated members, and optionally, (iii) a weight is expressed as a product of factors assigned to the edges along the path from the member m to n in the tree. The weighting system proposed by Gotoh (*112*) satisfies all these conditions, and is very easy to calculate requiring only O(N) time. The last condition (iii) makes profile-based operations feasible, which can save computation time by a factor of several orders of magnitude when N is large (*40, 112*).

An evolutionary tree is usually obtained from a given multiple alignment, the alignment is obtained by using pair weights, and the pair weights are calculated from the evolutionary tree. Thus the

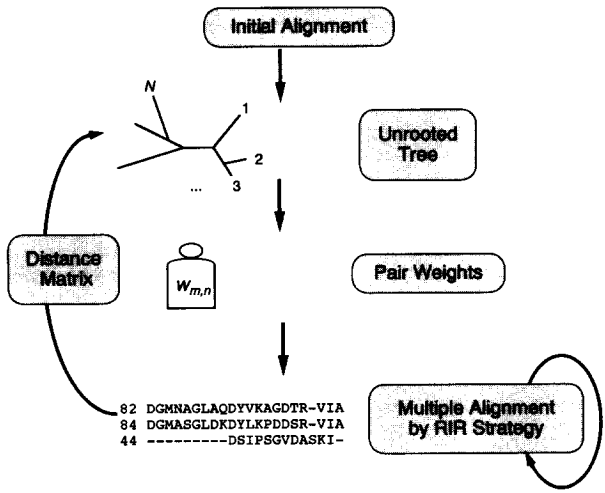

Fig. 9. Schematic representation of the DNR strategy. Slightly modified from Fig. 9 in ref. *14*.

multiple alignment, tree, and pair weights are mutually interdependent. Gotoh (*14*) proposed to keep consistency of this triad by iteration. The idea is essentially the same as that used in iterative tree reconstruction methods by Hogeweg and Hesper (*87*) and Corpet (*93*). Because the alignment is obtained by an RIR strategy, the overall protocol is outlined as doubly nested iterative loops (Fig. 9). This "doubly nested randomized iterative strategy" or DNR method has proven to be one of the best among all currently available multiple alignment methods, as discussed in more detail in Section IV.

5. *Stochastic Methods*

In the area of mathematical programming, various stochastic algorithms have been used to tackle hard optimization problems. Since multiple alignment is also formulated as a combinatorial optimization problem, such stochastic algorithms might be useful. Though reports are still sparse in the literature, applications of two representative stochastic optimization methods, simulated annealing and genetic algorithms are briefly introduced below.

A) Simulated annealing

Simulated annealing (SA) takes advantage of the analogy between statistical mechanics of the physical annealing process to solids and combinatorial optimization in complex systems (*113*). The process consists of a series of two kinds of operations, (i) change in the current state and (ii) acceptance or rejection of the change. Unlike iterative refinement strategies, which are hill-climbing approaches, locally unfavorable changes are accepted at a certain probability. The probability is controlled by the "temperature" and the energy gap between the old and new states. In multiple alignment, each state change is realized by a move of a gap or a block of gaps (*114–116*). The energy is calculated in the same way as an SP score (the sign is inverted), but any other scoring system may be appropriate. The cooling schedule is the most critical part for good performance. SA is an established method for alignment of protein 3D structures (*117–119*). As a tool for multiple alignment, however, it takes too much computation time, and is practical only if powerful computers are available (*115*). It seems most useful for fine tuning of near-optimal alignments (*120*) or mixed alignment of sequences and structures (*121*).

B) Genetic algorithms

Genetic algorithm (GA) is another popular stochastic method for solving complex optimization problems (*122*). It simulates a breeding process, involving mutations, crossovers, and selection. Each generation is composed of a population of a fixed size of, say, 100 individuals (alignments in our case). These individuals are evaluated by their fitness and those with higher fitness have better chance to produce their offspring. Various settings of mutations and crossovers as well as scoring systems are possible. Tajima (*123*) and Notredame and Higgins (*124*) took a rather straightforward setting, where a mutation corresponds to a move of a gap as in the case of SA, a crossover to a recombination between alignments at a consistent cut site (Section III-2C), and fitness to a WSP score. On the other hand, the setting of Ishikawa *et al.* (*125*) is a natural extension of the RIR strategy. Possibly the greatest merit of stochastic methods including GA and SA is the flexibility, particularly in the scoring system. Notredame *et al.* (*126*) applied GA to alignment of RNA sequences considering conserved features in both primary and secondary structure levels. Notredame *et al.* (*127*) recently proposed a unique objective function, COFFEE (Consistency based Objective Function For alignmEnt Evaluation) which measures the goodness of a multiple alignment by overall consistency between induced pairwise alignments (the projections onto (m, n) planes) and optimal pairwise alignments of the corresponding members. COFFEE seems particularly useful when structure-derived pairwise alignments are included in the reference library. Although GA is generally slower than RIR strategies, it well fits parallelization (*123, 125*). The flexibility of GA will facilitate its wider application in many fields of biological sequence analyses.

6. Reliable Regions in a Multiple Alignment

In Section III-2, several approaches to locate "robust" regions in a pairwise alignment have been discussed. Intersections of such robust regions may serve as anchor points which enable acceleration of multiple alignment. The problem considered by Gotoh (*128*) was motivated by a similar pragmatic question: given a (crude) multiple alignment $\mathcal{A}$, which parts are reliable? The approach was totally heuristic, following in the idea of local alignment methods (*129–131*).

```
H_0 := F := k := 0;
for l := 1 to L {
  H_l := MAX(H_{l-1} + S_l + G_l, 0);
  if (H_l > 0 and H_{l-1} = 0) L_k := l;
  if (H_l > MAX(θ, F)) {F := H_l; R_k := l;}
  if (F > θ and H_l < F - θ) {
    H_l := F := 0;
    k := k + 1;
  }
}
```

Fig. 10. The algorithm to locate reliably aligned regions between two groups of prealigned sequences (*128*). S_l and G_l are group-to-group WSP and gap penalty values, respectively, associated with the column position of l. Examples of the intervals $[L_h, R_h]$ ($h \in [1, k]$) thus obtained are shown in Fig. 12.

As in the DNR method (Section III-4C), we first obtain a rooted binary tree that represents the interrelationship among the sequences in $\mathcal{A}$. Then we proceed from the nodes nearest the leaves toward the root, just like the progressive multiple alignment methods (Section III-3). Each son of an internal node corresponds to a sequence or a group of similar sequences. A reliable region between these two (groups of) sequences is defined as that having a local alignment score greater than a given threshold θ.

The actual calculation resembles that described for calculation of the WSP score for aligned groups. Let S_l denote the WSP measure between the two groups of residues on the column position l, and G_l be the gap penalty associated with that position. We obtain S_l with one of the forms in Eq. 12, and G_l as described in Section II-3A. The running score H_l and the peak score F are obtained by an induction as shown in Fig. 10, where the threshold θ must be normalized according to the number of sequences in the groups and the weights assigned to individual sequences. The intervals $[L_k, R_k]$ are regarded as the robust regions. If a single robust region covers the entire alignment, the alignment can be fixed to behave as a single entity. The intersections among the robust regions observed for the two sons and those just obtained between the sons constitute the robust regions which are returned from the present node. The process proceeds recursively, and the regions associated with the root are regarded as the finally attained reliable cores.

The algorithm sketched above is implemented as a part of the multiple alignment program 'prrn' (Section V-1). Our experience with numerous groups of protein sequences has proven that the method is really very useful, although its theoretical background is relatively weak. For example, there is no justification for choosing a threshold value. Further theoretical work based on an elaborate probabilistic model would help to improve the credibility of the method and maximally widen the reliable ranges with minimal errors.

IV. EVALUATION OF ALIGNMENTS AND METHODS

The previous section discussed a plethora of alignment methods. How many and what length sequences can they align? How long must we wait for a result in front of a computer? How accurate are the alignments thus produced? To answer these questions, objective examinations are needed.

Probably the most important factor that discriminates various alignment methods is the number of sequences of reasonable lengths and diversities to be accepted. Roughly speaking, those methods which can align $N < 10$ sequences are small-sized, and those that can accept $N \geqq 10$ sequences are large. All exact methods and most of the combinatorial methods discussed in Section III-2 are classified in the former, while the progressive, iterative and stochastic methods (Section III-3–5) are classified in the latter. Since the number of sequences in a family of proteins or nucleic acids is growing rapidly, and more importantly, the accuracy of alignment tends to be improved by considering many sequences at a time (see below), the acceptable size is becoming more and more serious. Hence, the following discussions are mainly restricted to the large-sized methods.

1. Evaluation by Score

Since all the large-sized methods are heuristic in nature, they are not guaranteed to achieve global optimization. It is therefore meaningful to compare the scores obtained by different methods which target the same objective function, *e.g.* SP score. Hirosawa *et al.* (*110*) have made the most comprehensive tests along this line. They examined 22 methods including a progressive method and various

variants of RIR and BIR strategies. Average SP score and execution time were used for the evaluation. Two major conclusions were: (i) BIR strategies gave better scores on average than RIR strategies but were more prone to be trapped in local optima, and (ii) "tree-based" or "tree-restricted" RIR strategies showed relatively good performances. An obvious limitation of the score-based evaluation is its failure to compare methods targeting different objective functions. In addition, adequacy of the object function itself should be subject to tests.

2. *Evaluation by Structural References*

Currently the most satisfactory criterion for evaluating a sequence alignment is its closeness to the reference alignment derived from known protein 3D structures. Numerous authors have reported such comparisons, but most of them were fragmentary and lacked strong statistical support. With the rapid accumulation of known protein 3D structures, several databases of structural alignments have been developed (*132–137*). It is tempting to use these 3D alignments as the gold standards for evaluation of sequence alignments produced by various methods.

Vogt *et al.* (*13*) tested performance of various amino acid substitution matrices by querying how well pairwise sequence comparison by the use of a particular matrix reproduced the corresponding structural alignment. Inspired by this work, Gotoh (*14*) examined accuracy of sequence alignments obtained by several

TABLE I
Comparison of the Performance of Various Alignment Methods[a]

Method	Pairwise	Progressive	RIR without pair weights	DNR	RIR with pair weights
PWS	—	0.8 ± 1.2	3.5 ± 0.9***	6.7 ± 1.6***	7.1 ± 1.0***
CLW	5.7 ± 2.0**	—	2.7 ± 1.1**	5.9 ± 1.0***	6.3 ± 1.0***
RIO	8.5 ± 1.8***	2.8 ± 1.2*	—	3.2 ± 0.6***	3.6 ± 0.7***
DNR	13.2 ± 1.8***	7.5 ± 1.1***	4.7 ± 0.9***	—	0.4 ± 0.4
RIW	13.7 ± 1.9***	8.0 ± 1.2***	5.2 ± 1.0***	0.5 ± 0.4	—

[a] Difference in mean (upper right) or minimum (lower left) percentage of correctly matched residues obtained with the methods at the top and left corners. Statistical significance examined by the Wilcoxon matched-pairs signed rank test is indicated by: *$p < 0.05$, **$p < 0.01$, and ***$p < 0.001$. A positive value indicates that the top method performs better than the left method (upper right) and vice versa (lower left).

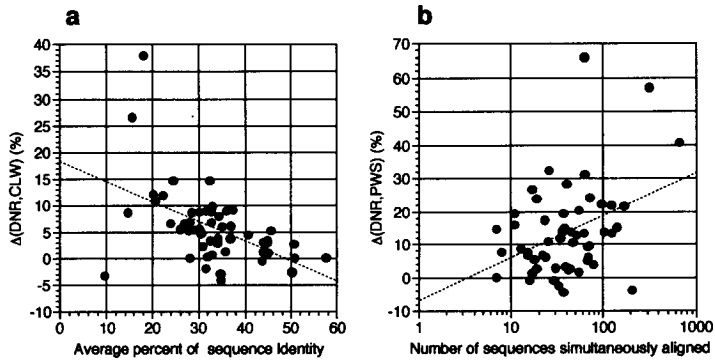

Fig. 11. Correlation between improvement in accuracy of alignment by the DNR method and average sequence identity (a) or number of sequences to be aligned (b). CLUSTAL W (*102*) (a) or pairwise method (b) (*26*) was used as the reference. The data were reproduced from ref. *14*.

methods, including pairwise, progressive and RIR strategies. Statistical significance in different performance was evaluated with non-parametric paired tests on 54 sets of protein families. Table I reproduces the summary of the test results, which show a clear order in performance: RIR methods which optimize WSP, RIW (*112*) and DNR (*14*) > RIR method which optimizes SP, RIO (*39, 40*) > progressive method, CLUSTAL W (*102*) > pairwise method (*26*). The difference in performance between DNR (or RIW) and CLUSTAL W or pairwise methods was more accentuated with a decrease in identities among sequences, and with an increase in the number of sequences to be aligned (Fig. 11). Thus DNR (or RIW) is preferable to the representative progressive method under generally difficult conditions. Although DNR ran a few times slower than CLUSTAL W, fine tuning of the codes has nearly halved the execution time compared to the original implementation (Gotoh, unpublished).

Although 3D structures of proteins are much better conserved than sequences, unequivocal alignment between distantly related homologous structures is still difficult (*138*). A few methods have been proposed to moderate potential influence of this ambiguity in structural alignments to qualification of sequence alignments. Gotoh (*14*) proposed to use only the regions consistent between two

different sets of structural alignments for the evaluation. Barton and Sternberg (*91*) and Notredame *et al.* (*127*) used a single set but only the regions that form secondary structures (helices and sheets), since ambiguity in structural alignments is largely due to conformational divergence in loop regions. McClure *et al.* (*139*) went farther; they evaluated a sequence alignment method by its ability to correctly reproduce the ordered series of motifs characteristic of a protein family. Similarly, the tests conducted by Briffeuil *et al.* (*140*) were mainly concerned with local alignments, while the references were drawn on structural bases.

These kinds of evaluations ought to be statistical in nature. Adjustment of one or a few parameter values can drastically affect the quality of the result in individual applications. Such an example will be shown in the next section. Unfortunately, appropriate sets of parameter values are usually unpredictable from sequences alone. It is also hard to predict the behavior of the solution when one or more outsiders stray into a family of sequences, or some members are suffering from artificial errors (frame-shift errors in particular). Future evaluation systems should co-evolve with the alignment methods themselves so that the degrees of reliability of individual regions in an alignment are predictable.

V. APPLICATIONS

1. Running Prrn

Prrn (Profile-based Randomized iterative Refinement method for alignment of Nucleotide or amino acid sequences) is a program that implements the RIR strategies seeking optimal WSP (*14, 112*). The program is written in ANSI-C programming language, has been tested on several UNIX operating systems, and is freely available for non-commercial uses. The source code is available through the Internet from ftp.genome.ad.jp/genomenet/saitama-cc. In principle, it can align hundreds of sequences of moderate lengths (up to about a thousand amino acids or nucleotides), but actual computation time strongly depends on the degree of sequence divergence, and lengths and distribution of gaps. Despite the name, the program is not suitable for alignment of structural nucleotide sequences such as rRNA, since no information about potential secondary structures is taken into account.

TABLE II
Main Options for the Prrn Program

Option	Argument	Range	Description
-a	Δ or Integer	0–2	Get the initial alignment by a successive method
-b	Δ or Integer	0–2	Get the initial alignment by a progressive method
-F	Integer	1–9	Output format
-H	Integer	≧ 0	Threshold value θ
-I	Integer	≧ 1	Maximum number of iterations for the outer loop
-l	Integer	≧ 0	Number of residues in each line of output
-m	String	File	Amino-acid substitution matrix
-o	String	File	Destination of the output
-O	Integer	0–2	Verbosity in output
-p	None		Set parameter values interactively
-R	Integer	≧ 0	Seed of the series of random numbers
-s	String	Directory	Where the input file is stored
-S	Integer	≧ 1	Number of repeated series of iterations
-t	None		Rearrange the output in the order of inferred tree topology
-u	Integer	≧ 0	Gap-extension penalty
-v	Integer	≧ 0	Gap-open penalty
-w	Integer	≧ 0	Bandwidth around the main diagonal

The standard format to run "prrn" is very simple:

% prrn [options] InputFile

The InputFile contains an alignment of sequences to be refined. If not specified, the program automatically judges the molecular species (DNA, RNA, or proteins). Two formats (sequential and native) are currently supported to represent the input alignment. The input alignment may be trivial, *e.g.* left justified, tailing null codes being dispensable. At that time, the user is advised to set an option "-a*N*" or "-b*N*" (*N* may be empty or an integer) to initiate the iterative refinement from a successively or progressively aligned state. Several options most frequently used are listed in Table II. By default, the DNR strategy is chosen, but the option set "-S*N* -I1" ($N > 1$) evokes the RIW without reevaluation of the initial pair weights.

The result goes to "stdout" unless specified by "-oOutputFile". Several formats can be culled by option "-F*N*", which enables smooth interface with other useful programs such as PHD

(*141*) and DSC (*142*) for protein secondary structure prediction, phyln (Gotoh, distributed with prrn), PHYLIP (*143*) and MOLPHY (*144*) for phylogenetic reconstruction, and CLUSTAL X (*145*) for better visualization and further manual edition of the alignment.

2. *An Example*

Figure 12 shows the distribution of sequence conservation along the alignment of RecA-related proteins obtained by prrn. This shows a demonstrative example, in which proper setting of options had definitive effects on the quality of the result. The 23 sequences drawn from eubacterial RecA, archae RadA, and eukaryotic Rad51 families are nearly the same as those recently examined by Brocchieri and Karlin (*146*). Since they are missing in the current SWISS-PROT database, two sequences (RADA_HALHA and RECA_SYNP7) were replaced by other homologues (PIR:A69374 and RECA_SYNP2). A long insertion in RADA_METJA relative

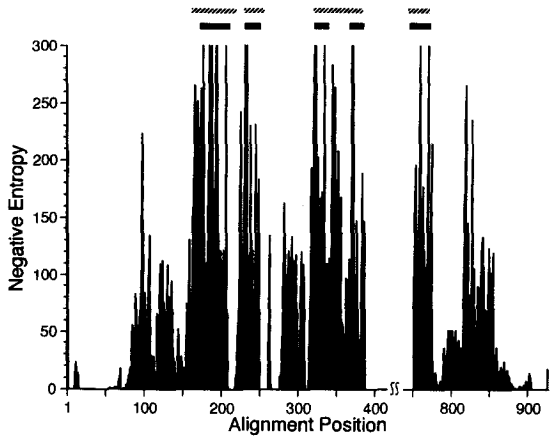

Fig. 12. A conservation profile for twenty-three sequences in recA/rad51 families. The sequences were multiply aligned by the DNR method, and the degree of sequence conservation at each column position was evaluated by the negative entropy as formulated in Fig. 2d. The reliably aligned regions obtained by the algorithm outlined in Fig. 10 are indicated by black bars. The blocks containing seven conserved motifs which were pointed out by Brocchieri and Karlin (*146*) are indicated by hatched bars.

to other members was retained in our analysis, whereas Brocchieri and Karlin appeared to have removed it beforehand. By the default setting, prrn, as well as CLUSTAL W, demonstrated none of the seven conserved motifs used by Brocchieri and Karlin as the indicators of correct alignment. With "-w120" option, which widens the bandwidth around the major diagonals to be scanned in each group-to-group alignment (Fig. 6a), a correct alignment emerged after the second cycle of the outer loop of the DNR strategy. When the initial alignment was obtained by a progressive method with the options of "-w120-b2", the seven motifs appeared before the first cycle. The "-b2" option forces the program to construct the initial guide tree based on distances calculated from pairwise alignments between all combinations of members, whereas the distances are thrown up by a much simpler method in the default setting. This observation indicates one pitfall of incautious use of a particular program, not only for individual applications but also for evaluation of the method.

VI. FESS RELATIONSHIPS AND MULTIPLE ALIGNMENT

Multiple alignment is tightly interconnected with phylogenetic reconstruction, structural prediction, and functional featuring, each of which might deserve a voluminous chapter or book. Currently multiple alignment is mostly used as a preprocessing tool for investigations of these FESS relationships. However, tighter mutual connection between multiple alignment and the FESS relationships will be indispensable for concerted improvement in quality of all of these sequence-based predictions. In this section, we cast a quick glance at individual problems within the scope of their relevance to multiple alignment.

1. Phylogenetic Reconstruction

Almost all methods for phylogenetic reconstruction from molecular sequence data (*147–151*) rely on a given multiple alignment. Hence, quality of an alignment significantly affects the reliability of the inferred evolutionary tree (*152*). As exemplified in Fig. 12, the degrees of conservation/variation at individual sites along a multiple alignment are extensively variable. Some recent phylogenetic reconstruction methods take site variation into account (*153*). Mul-

tiple alignment is often considered a "given" before phylogenetic inference, but the interdependency between them is very tight. In fact, many multiple alignment methods use a phylogenetic tree as the guide of progressive iteration or to calculate the objective function, as already shown in the preceding section. A serious problem criticized by Lake (*154*) is that the initial tentative tree topology can dominate the final result obtained through these tree-dependent alignment methods. Ideally, the tree and alignment should be obtained simultaneously, but the huge amount of necessary time and space render such approaches impractical (*155, 156*). The best heuristics at this moment appear to be iterative tree-reconstruction strategies (*87, 93*) combined with optimization of alignment at each iterative stage, just as the DNR method has adopted but with a more advanced tree-reconstruction method. Active use of multifurcating tree topologies might be effective to refine an alignment when branching orders are indefinite.

2. Functional Featuring

With the progress in genome projects, assigning a function to each potential gene (ORF) is an urgent problem (*157, 158*). Homology search has been proven the most effective tool for this purpose. Many investigations have shown that multiple alignment (or profile or HMM derived therefrom), when used as a query, significantly improves sensitivity and specificity of a search (*159*). The search and construction of a query may be performed iteratively (*160–162*). Statistical theories (*163, 164*) on the extreme value distribution of search results have given credibility to weak signals detected by these procedures. The signatures discovered by these approaches are derived from relatively short regions conserved across divergent homologues. Hence, precise multiple alignment over entire sequences is not the major concern. Nevertheless, global multiple alignment sometimes finds conserved motifs which are missed by local methods (*139, 140, 146*). Moreover, global or semi-global alignments can be more informative than local alignments, when overall protein 3D structures are conserved without characteristic motifs. In one extreme, the functionally most important regions coincide with evolutionarily most variable ones, as observed in proteins involved in some defensive systems, such as immunological and detoxification systems (*165, 166*).

3. *Prediction of Protein Structure*

Accurate sequence alignment is a prerequisite for reliable homology modeling of proteins (*167*) on the basis of a known 3D structure(s) of a family member(s) (*168*). Multiple sequence alignment is one of a few approaches which have been proven effective to improve accuracy of alignments. Another approach, known as "threading", directly matches a sequence and a structure. Threading has been mostly used for "fold-recognition", *i.e.*, for finding a structural class to which the given sequence might belong. It is still uncertain to what extent structural information helps to improve alignment accuracy beyond the well-known observations that indels rarely occur in helical and sheet regions (*169, 170*).

Accumulating reports have shown that the accuracy in prediction of protein secondary structure is markedly improved by the use of multiple sequence alignments (*141, 142, 171–175*). The factors responsible for the improved accuracy are (i) statistical gain of signal to noise ratio, (ii) periodicity in conservation patterns, (iii) distribution of gaps, and (iv) variable evolutionary constraints on amino-acid substitution patterns with different secondary-structural states. In the artificial neural network methods (*141*) and nearest-neighbor method (*173*), these factors contribute in a complex fashion, which prohibits anatomical analyses of relative contributions of these and other factors to the improvement. The linear discrimination method (*142*) is transparent in this respect, but seemingly difficult in incorporating the fourth factor. On the other hand, the method of Mehta *et al.* (*174*) and the HMM formulation of Goldman *et al.* (*175*) put stress on the fourth factor. Instead of the different stress of points, these methods have generally achieved prediction accuracies above 70%. Therefore, there seems to be room for further improvement in prediction methods and eventual attainment of more than 85% residue-base accuracy as suggested by Rost *et al.* (*172*).

Sometimes similarity in predicted secondary structures and hydrophobicity (or solvent accessibility) profiles has been incorporated into the score of a sequence alignment (*112, 176, 177*). Correct locations of helices conserved between distantly related bacterial and mammalian cytochrome P450 families were recognized (*166*) only with the help of predicted secondary structure information (*178*). Critical evaluation of the effects of such predicted local

environmental factors on quality of alignments has not yet been carried out. Considering the recent success in higher prediction rates, we are convinced that predicted propensities in secondary structure formation and burial status will facilitate reliable alignment among distantly related families of proteins (*179*).

4. Generalized Alignment

So far, we have discussed the alignment of either nucleotide or amino acid sequences. This limitation of the same kind of molecular species may be taken away so long as the collinearity holds for the sequences to be aligned. An obvious example is alignment between a protein-coding DNA sequence and a related protein sequence. This generalized or "mixed" alignment has an advantage of detecting possible frame shifts in the DNA sequence (*180, 181*). When the DNA is a eukaryotic genomic sequence that contains a segmented gene, such a mixed alignment helps to delineate the precise exon/intron boundaries of the gene (*182, 183*).

Threading is also regarded as a mixed alignment between a sequence and a structure. For threading, like secondary structure formation in RNAs, we must consider non-local interactions, which renders the problem extremely hard (*184*). With the 3D-1D profiles (*185*) that consider only local structural features, this difficulty is avoidable at the expense of potential loss of information.

Simultaneous consideration of multiple sequences and/or structures would certainly improve the quality of mixed alignments. Because of the computational difficulty, however, only a few attempts have been reported. Gotoh (*186*) has developed a program to predict the precise exon/intron organization of a eukaryotic gene through alignment between the genomic DNA sequence and a generalized protein profile. Iterative exon prediction and profile construction for a set of homologous genes significantly improved the overall accuracy of predicted organizations of individual genes. Several authors have reported methods for multiple structural alignment (*187, 188*). Taylor (*189*) has proposed a method for multiple threading. Although these approaches are still in their infancy, generalized alignment of many sequences and structures will become a most useful tool for studies on FESS relationships of a family of proteins and polynucleotides.

SUMMARY

Elucidation of interrelationships among sequence, structure, function, and evolution (FESS relationships) of a family of genes or gene products is a central theme of modern molecular biology. Multiple sequence alignment has been proven to be a powerful tool for many fields of studies such as phylogenetic reconstruction, illumination of functionally important regions, and prediction of higher order structures of proteins and RNAs. However, it is far too trivial to automatically construct a multiple alignment from a set of related sequences. A variety of methods for solving this computationally difficult problem are reviewed. Several important applications of multiple alignment for elucidation of the FESS relationships are also discussed.

For a long period, progressive methods have been the only practical means to solve a multiple alignment problem of appreciable size. This situation is now changing with the development of new techniques including several classes of iterative methods. Today's progress in multiple sequence alignment methods has been made by the multidisciplinary endeavors of mathematicians, computer scientists, and biologists in various fields including biophysicists in particular. The ideas are also originated from various backgrounds, pure algorithmics, statistics, thermodynamics, and others. The outcomes are now enjoyed by researchers in many fields of biological sciences.

In the near future, generalized multiple alignment may play a central role in studies of FESS relationships. The organized mixture of knowledge from multiple fields will ferment to develop fruitful results which would be hard to obtain within each area. I hope this review provides a useful information resource for future development of theory and practice in this rapidly expanding area of bioinformatics.

Acknowledgments

I am very grateful to Professor Akiyoshi Wada who recommended preparation of this review. This work was partly supported by Grants-in-Aid from the Ministry of Education, Science, Sports and Culture of Japan.

REFERENCES

1 M. Kimura, "The Neutral Theory of Molecular Evolution," Cambridge University Press, Cambridge (1983).
2 D. Sankoff and J. B. Kruskal, "Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison," Addison-Wesley, New York (1983).
3 M. S. Waterman, "Introduction to Computational Biology," Chapman & Hall, London (1995).
4 R. F. Doolittle (ed.), "Methods in Enzymology," Vol. 183 (1990).
5 R. F. Doolittle (ed.), "Methods in Enzymology," Vol. 266 (1996).
6 M. J. E. Sternberg (ed.), "Protein Structure Prediction," Oxford University Press, Oxford (1996).
7 M. J. Bishop and C. J. Rawlings (eds.), "DNA and Protein Sequence Analysis," Oxford University Press, Oxford (1997).
8 D. Gusfield, "Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology," Cambridge University Press, Cambridge (1997).
9 R. Durbin, S. Eddy, A. Krogh, and G. Mitchison, "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids," Cambridge University Press, Cambridge (1998).
10 M. Dayhoff (ed.), "Atlas of Protein Sequence and Structure," Vol. 5, Suppl. 3, National Biomedical Research Foundation, Washington, D.C. (1978).
11 K. Tomii and M. Kanehisa, *Protein Eng.*, **9**, 27 (1996).
12 G. H. Gonnet, M. A. Cohen, and S. A. Benner, *Science*, **256**, 1443 (1992).
13 G. Vogt, T. Etzold, and P. Argos, *J. Mol. Biol.*, **249**, 816 (1995).
14 O. Gotoh, *J. Mol. Biol.*, **264**, 823 (1996).
15 S. Pascarella and P. Argos, *J. Mol. Biol.*, **224**, 461 (1992).
16 E. G. Shpaer, M. Robinson, D. Yee, J. D. Candlin, R. Mines, and T. Hunkapiller, *Genomics*, **38**, 179 (1996).
17 S. A. Benner, M. A. Cohen, and G. H. Gonnet, *J. Mol. Biol.*, **229**, 1065 (1993).
18 M. S. Waterman, *Bull. Math. Biol.*, **46**, 473 (1984).
19 S. B. Needleman and C. D. Wunsch, *J. Mol. Biol.*, **48**, 443 (1970).
20 P. H. Sellers, *J. Algorithms*, **1**, 359 (1980).
21 M. S. Waterman, T. F. Smith, and W. A. Beyer, *Adv. Math.*, **20**, 367 (1976).
22 T. F. Smith, M. S. Waterman, and W. M. Fitch, *J. Mol. Evol.*, **18**, 38 (1981).
23 D. Sankoff, *Proc. Natl. Acad. Sci. USA*, **69**, 4 (1972).
24 P. H. Sellers, *SIAM J. Appl. Math.*, **26**, 787 (1974).
25 R. A. Wagner and M. J. Fischer, *J. Assoc. Comp. Mach.*, **21**, 168 (1974).
26 O. Gotoh, *J. Mol. Biol.*, **162**, 705 (1982).
27 O. Gotoh, *Bull. Math. Biol.*, **52**, 359 (1990).
28 X. Huang, *Comput. Appl. Biosci.*, **10**, 227 (1994).
29 W. Miller and E. W. Myers, *Bull. Math. Biol.*, **50**, 97 (1988).
30 Z. Galil and R. Giancarlo, *Theor. Comp. Sci.*, **64**, 107 (1989).
31 L. Allison, C. S. Wallace, and C. N. Yee, *J. Mol. Evol.*, **35**, 77 (1992).
32 A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "Data Structures and Algorithms," Addison-Wesley, Reading (1983).
33 S. F. Altschul and B. W. Erickson, *Bull. Math. Biol.*, **48**, 603 (1986).
34 P. Taylor, *Nucl. Acids Res.*, **12**, 447 (1984).
35 D. S. Hirshberg, *Commun. Assoc. Comput. Mach.*, **18**, 341 (1975).
36 E. W. Myers and W. Miller, *Comp. Appl. Biosci.*, **4**, 11 (1988).
37 K.-M. Chao, R. C. Hardison, and W. Miller, *J. Comput. Biol.*, **1**, 271 (1994).
38 M. Zuker, *J. Mol. Biol.*, **221**, 403 (1991).

39 O. Gotoh, *Comp. Appl. Biosci.*, **9**, 361 (1993).
40 O. Gotoh, *Comp. Appl. Biosci.*, **10**, 379 (1994).
41 M. Gribskov, A. D. McLachlan, and D. Eisenberg, *Proc. Natl. Acad. Sci. USA*, **84**, 4355 (1987).
42 D. Sankoff, *SIAM J. Appl. Math.*, **78**, 35 (1975).
43 S. F. Altschul, *J. Theor. Biol.*, **138**, 297 (1989).
44 M. L. Fredman, *Bull. Math. Biol.*, **46**, 553 (1984).
45 M. Murata, J. S. Richardson, and J. L. Sussman, *Proc. Natl. Acad. Sci. USA*, **82**, 3073 (1985).
46 O. Gotoh, *J. Theor. Biol.*, **121**, 327 (1986).
47 D. Sankoff, R. J. Cedergren, and G. Lapalme, *J. Mol. Evol.*, **7**, 133 (1976).
48 H. Imai and T. Ikeda, "Genome Informatics Workshop," Vol. 6, ed. by M. Hagiya *et al.*, Universal Academy, Tokyo, p. 9 (1995).
49 J. W. Fickett, *Nucl. Acids Res.*, **12**, 175 (1984).
50 E. Ukkonen, *Inf. Control.*, **64**, 100 (1985).
51 H. Carrillo and D. Lipman, *SIAM J. Appl. Math.*, **48**, 1073 (1988).
52 J. L. Spouge, *SIAM J. Appl. Math.*, **49**, 1552 (1989).
53 J. L. Spouge, *Compt. Appl. Biosci.*, **7**, 1 (1991).
54 P. E. Hart, N. J. Nilsson, and B. Raphael, *IEEE Trans. Syst. Sci. Cybern.*, **4**, 100 (1968).
55 S. Araki, M. Goshima, S.-i. Mori *et al.*, "Genome Informatics Workshop," Vol. 4, ed. by T. Takagi *et al.*, Universal Academy, Yokohama, p. 94 (1993).
56 D. J. Lipman, S. F. Altschul, and J. D. Kececioglu, *Proc. Natl. Acad. Sci. USA*, **86**, 4412 (1989).
57 S. K. Gupta, J. D. Kececioglu, and A. A. Schaffer, *J. Comput. Biol.*, **2**, 459 (1995).
58 T. Ikeda and H. Imai, *Theor. Comput. Sci.*, **210**, 341 (1999).
59 H. Kobayashi and H. Imai, "Genome Informatics," Vol. 9, ed. by S. Miyano and T. Takagi, Universal Academy, Tokyo, p. 120 (1998).
60 E. Sobel and H. M. Martinez, *Nucl. Acids Res.*, **14**, 363 (1986).
61 L. J. Korn, C. L. Queen, and M. N. Wegman, *Proc. Natl. Acad. Sci. USA*, **74**, 4401 (1977).
62 M. Santibánez and K. Rohde, *Comp. Appl. Biosci.*, **3**, 111 (1987).
63 M. Vingron and P. Argos, *Comp. Appl. Biosci.*, **5**, 115 (1989).
64 J.-P. Dumas and J. Ninio, *Nucl. Acids Res.*, **10**, 197 (1982).
65 W. J. Wilbur and D. J. Lipman, *Proc. Natl. Acad. Sci. USA*, **80**, 726 (1983).
66 G. D. Schuler, S. F. Altschul, and D. J. Lipman, *Proteins*, **9**, 180 (1991).
67 R. Staden, *Nucl. Acids Res.*, **10**, 2951 (1982).
68 M. Vingron and P. Argos, *J. Mol. Biol.*, **218**, 33 (1991).
69 M. Vingron and P. A. Pevzner, *Adv. Appl. Math.*, **16**, 1 (1995).
70 E. Depiereux and E. Feytmans, *Comp. Appl. Biosci.*, **8**, 501 (1992).
71 B. Morgenstern, A. Dress, and T. Werner, *Proc. Natl. Acad. Sci. USA*, **93**, 12098 (1996).
72 M. S. Johnson and R. F. Doolittle, *J. Mol. Evol.*, **23**, 267 (1986).
73 M. S. Waterman, *Nucl. Acids Res.*, **14**, 9095 (1986).
74 O. Gotoh, *Bull. Math. Biol.*, **52**, 509 (1990).
75 W. Miller, *Comput. Appl. Biosci.*, **9**, 169 (1993).
76 M. Vingron and P. Argos, *Protein Eng.*, **3**, 565 (1990).
77 K.-M. Chao, R. C. Hardison, and W. Miller, *Comput. Appl. Biosci.*, **9**, 387 (1993).
78 H. T. Mevissen and M. Vingron, *Protein Eng.*, **9**, 127 (1996).
79 M. J. Bishop and E. A. Thompson, *J. Mol. Biol.*, **190**, 159 (1986).
80 J. L. Thorne, H. Kishino, and J. Felsenstein, *J. Mol. Evol.*, **33**, 114 (1991).

81 J. L. Thorne, H. Kishino, and J. Felsenstein, *J. Mol. Evol.*, **34**, 3 (1992).
82 M. Q. Zhang and T. G. Marr, *J. Theor. Biol.*, **174**, 119 (1995).
83 S. Miyazawa, *Protein Eng.*, **8**, 999 (1995).
84 U. Tönges, S. W. Perrey, J. Stoye, and A. W. M. Dress, *Gene*, **172**, GC33 (1996).
85 G. Brinkmann, A. W. M. Dress, S. W. Perrey, and J. Stoye, "Mathematical Programming," Vol. 79, ed. by T. M. Liebling and D. de Werra, Elsevier, Amsterdam, p. 71 (1997).
86 J. Stoye, *Gene*, **211**, GC45 (1998).
87 P. Hogeweg and B. Hesper, *J. Mol. Evol.*, **20**, 175 (1984).
88 M. S. Waterman and M. D. Perlwitz, *Bull. Math. Biol.*, **46**, 567 (1984).
89 D.-F. Feng and R. F. Doolittle, *J. Mol. Evol.*, **25**, 351 (1987).
90 W. R. Taylor, *J. Mol. Evol.*, **28**, 161 (1988).
91 G. J. Barton and M. J. E. Sternberg, *J. Mol. Biol.*, **198**, 327 (1987).
92 R. F. Smith and T. F. Smith, *Proc. Natl. Acad. Sci. USA*, **87**, 118 (1990).
93 F. Corpet, *Nucl. Acids Res.*, **16**, 10881 (1988).
94 D. G. Higgins and P. M. Sharp, *Comp. Appl. Biosci.*, **5**, 151 (1989).
95 J. Hein, *Mol. Biol. Evol.*, **6**, 649 (1989).
96 S. C. Chan, A. K. C. Wong, and D. K. Y. Chiu, *Bull. Math. Biol.*, **54**, 563 (1992).
97 D.-F. Feng and R. F. Doolittle, *Methods Enzymol.*, **266**, 368 (1996).
98 D. G. Higgins, J. D. Thompson, and T. J. Gibson, *Methods Enzymol.*, **266**, 383 (1996).
99 J. Hein, *Mol. Biol. Evol.*, **6**, 669 (1989).
100 O. Trelles, M. A. Andrade, A. Valencia, E. L. Zapata, and J. M. Carazo, *Bioinformatics*, **14**, 439 (1998).
101 W. Bains, *Nucl. Acids Res.*, **14**, 159 (1986).
102 J. D. Thompson, D. G. Higgins, and T. J. Gibson, *Nucl. Acids Res.*, **22**, 4673 (1994).
103 W. R. Taylor, *Gene*, **1**, GC27 (1995).
104 A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler, *J. Mol. Biol.*, **235**, 1501 (1994).
105 P. Baldi, Y. Chauvin, T. Hunkapiller, and M. A. McClure, *Proc. Natl. Acad. Sci. USA*, **91**, 1059 (1994).
106 L. R. Rabiner, *Proc. IEEE*, **77**, 257 (1989).
107 R. Hughey and A. Krogh, *Comput. Appl. Biosci.*, **12**, 95 (1996).
108 S. Subbiah and S. C. Harrison, *J. Mol. Biol.*, **209**, 539 (1989).
109 M. P. Berger and P. J. Munson, *Comp. Appl. Biosci.*, **7**, 479 (1991).
110 M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa, *Comp. Appl. Biosci.*, **11**, 13 (1995).
111 S. F. Altschul, R. J. Carroll, and D. J. Lipman, *J. Mol. Biol.*, **207**, 647 (1989).
112 O. Gotoh, *Comp. Appl. Biosci.*, **11**, 543 (1995).
113 S. Kirkpatrick, C. D. J. Gelatt, and M. P. Vecchi, *Science*, **220**, 671 (1983).
114 A. V. Lukashin, J. Engelbrecht, and S. Brunak, *Nucl. Acids Res.*, **20**, 2511 (1992).
115 M. Ishikawa, T. Toya, M. Hoshida, K. Nitta, A. Ogiwara, and M. Kanehisa, *Comput. Appl. Biosci.*, **9**, 267 (1993).
116 J. Kim, S. Pramanik, and M. J. Chung, *Comput. Appl. Biosci.*, **10**, 419 (1994).
117 A. Šali and T. L. Blundell, *J. Mol. Biol.*, **212**, 403 (1990).
118 L. Holm and C. Sander, *J. Mol. Biol.*, **233**, 123 (1993).
119 M. S. Johnson, J. P. Overington, and T. L. Blundell, *J. Mol. Biol.*, **231**, 735 (1993).
120 M. Hirosawa, M. Hoshida, M. Ishikawa, and Y. Toya, *Comput. Appl. Biosci.*, **9**, 161 (1993).
121 A. Godzik and J. Skolnick, *Comput. Appl. Biosci.*, **10**, 587 (1994).

122 D. E. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, New York (1989).
123 K. Tajima, "Genome Informatics Workshop," Vol. 4, ed. by T. Takagi *et al.*, Universal Academy, Yokohama, p. 183 (1993).
124 C. Notredame and D. G. Higgins, *Nucl. Acids Res.*, **24**, 1515 (1996).
125 M. Ishikawa, T. Toya, Y. Totoki, and A. Konagaya, "Genome Informatics Workshop," Vol. 4, ed. by T. Takagi *et al.*, Universal Academy, Yokohama, p. 84 (1993).
126 C. Notredame, E. A. O'Brien, and D. G. Higgins, *Nucl. Acids Res.*, **25**, 4570 (1997).
127 C. Notredame, L. Holm, and D. G. Higgins, *Bioinformatics*, **14**, 407 (1998).
128 O. Gotoh, "Genome Informatics Workshop," Vol. 4, ed. by T. Takagi *et al.*, Universal Academy, Yokohama, p. 109 (1993).
129 T. F. Smith and M. S. Waterman, *J. Mol. Biol.*, **147**, 195 (1981).
130 W. B. Goad and M. I. Kanehisa, *Nucl. Acids Res.*, **10**, 247 (1982).
131 O. Gotoh, *Comp. Appl. Biosci.*, **3**, 17 (1987).
132 A. Šali and J. P. Overington, *Protein Sci.*, **3**, 1582 (1994).
133 A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, *J. Mol. Biol.*, **247**, 536 (1995).
134 S. Pascarella, F. Milpetz, and P. Argos, *Protein Eng.*, **9**, 249 (1996).
135 L. Holm and C. Sander, *Nucl. Acids Res.*, **24**, 206 (1996).
136 K. Mizuguchi, C. M. Deane, T. L. Blundell, and J. P. Overington, *Protein Sci.*, **7**, 2469 (1998).
137 R. Sowdhamini, D. F. Burke, J. F. Huang *et al.*, *Structure*, **6**, 1087 (1998).
138 A. Godzik, *Protein Sci.*, **5**, 1325 (1996).
139 M. A. McClure, T. K. Vasi, and W. M. Fitch, *Mol. Biol. Evol.*, **11**, 571 (1994).
140 P. Briffeuil, G. Baudoux, C. Lambert *et al.*, *Bioinformatics*, **14**, 357 (1998).
141 B. Rost and C. Sander, *J. Mol. Biol.*, **232**, 584 (1993).
142 R. King and M. J. E. Sternberg, *Protein Sci.*, **5**, 2298 (1996).
143 J. Felsenstein, PHYLIP Ver. 3.5c, University of Washington, Seattle (1992).
144 J. Adachi and M. Hasegawa, MOLPHY Ver. 2.3, *Comput. Sci. Monogr.*, **28**, 1 (1996).
145 J. D. Thompson, T. J. Gibson, F. Plewniak, F. Jeanmougin, and D. G. Higgins, *Nucl. Acids Res.*, **25**, 4876 (1997).
146 L. Brocchieri and S. Karlin, *J. Mol. Biol.*, **276**, 249 (1998).
147 J. Felsenstein, *Annu. Rev. Genet.*, **22**, 521 (1988).
148 M. Nei, *Annu. Rev. Genet.*, **30**, 371 (1996).
149 D. L. Swofford, G. J. Olsen, P. J. Waddell, and D. M. Hillis, "Molecular Systematics," 2nd ed., ed. by D. M. Hillis, C. Moritz, and B. K. Mable, Sunderland, Sinauer, p. 407 (1996).
150 N. Saitou, *Methods Enzymol.*, **266**, 427 (1996).
151 M. Hasegawa and H. Kishino, "Bunshi Keitougaku," Iwanami, Tokyo (1996) (in Japanese).
152 D. A. Morrison and J. T. Ellis, *Mol. Biol. Evol.*, **14**, 428 (1997).
153 Z. Yang, *J. Mol. Evol.*, **39**, 306 (1994).
154 J. A. Lake, *Mol. Biol. Evol.*, **8**, 378 (1991).
155 L. Allison and C. S. Wallace, *J. Mol. Evol.*, **39**, 418 (1994).
156 G. Mitchison and R. Durbin, *J. Mol. Evol.*, **41**, 1139 (1995).
157 P. Bork and E. Koonin, *Nature Genet.*, **18**, 313 (1998).
158 T. F. Smith, *Trends Genet.*, **14**, 291 (1998).
159 J. Park, K. Karplus, C. Barrett *et al.*, *J. Mol. Biol.*, **284**, 1201 (1998).
160 T.-M. Yi and E. S. Lander, *Protein Sci.*, **3**, 1315 (1994).
161 A. F. Neuwald, J. S. Liu, D. J. Lipman, and C. E. Lawrence, *Nucl. Acids Res.*, **25**,

1665 (1997).
162 S. F. Altschul, T. L. Madden, A. A. Schaffer *et al.*, *Nucl. Acids Res.*, **25**, 3389 (1997).
163 S. Karlin and S. F. Altschul, *Proc. Natl. Acad. Sci. USA*, **87**, 2264 (1990).
164 S. Karlin and S. F. Altschul, *Proc. Natl. Acad. Sci. USA*, **90**, 5873 (1993).
165 A. L. Hughes and M. Nei, *Nature*, **335**, 167 (1988).
166 O. Gotoh, *J. Biol. Chem.*, **267**, 83 (1992).
167 M. A. S. Saqi, P. A. Bates, and M. J. E. Sternberg, *Protein Eng.*, **5**, 305 (1992).
168 T. L. Blundell, B. L. Sibanda, M. J. E. Sternberg, and J. M. Thornton, *Nature*, **326**, 347 (1987).
169 A. M. Lesk, M. Levitt, and C. Chothia, *Protein Eng.*, **1**, 77 (1986).
170 Z. Y. Zhu, A. Šali, and T. L. Blundell, *Protein Eng.*, **5**, 43 (1992).
171 J. M. Levin, S. Pascarella, P. Argos, and J. Garnier, *Protein Eng.*, **6**, 849 (1996).
172 B. Rost, C. Sander, and R. Schneider, *J. Mol. Biol.*, **235**, 13 (1994).
173 A. A. Salamov and V. V. Solovyev, *J. Mol. Biol.*, **247**, 11 (1995).
174 P. K. Mehta, J. Heringa, and P. Argos, *Protein Sci.*, **4**, 2517 (1995).
175 N. Goldman, J. L. Thorne, and D. T. Jones, *J. Mol. Biol.*, **263**, 196 (1996).
176 F. Fischel-Ghodsian, G. Mathiowitz, and T. F. Smith, *Protein Eng.*, **3**, 577 (1990).
177 L. H. Bell, J. R. Coggins, and E. J. Milner-White, *Protein Eng.*, **6**, 683 (1993).
178 J.-F. Gibrat, J. Garnier, and B. Robson, *J. Mol. Biol.*, **198**, 425 (1987).
179 B. Rost, R. Schneider, and C. Sander, *J. Mol. Biol.*, **270**, 471 (1997).
180 H. Peltola, H. Soderlund, and E. Ukkonen, *Nucl. Acids Res.*, **14**, 99 (1986).
181 D. J. States and D. Botstein, *Proc. Natl. Acad. Sci. USA*, **88**, 5518 (1991).
182 X. Huang and J. Zhang, *Comput. Appl. Biosci.*, **12**, 497 (1996).
183 M. S. Gelfand, A. A. Mironov, and P. A. Pevzner, *Proc. Natl. Acad. Sci. USA*, **93**, 9061 (1996).
184 R. H. Lathrop, *Protein Eng.*, **7**, 1059 (1994).
185 J. U. Bowie, R. Lüthy, and D. Eisenberg, *Science*, **253**, 164 (1991).
186 O. Gotoh, *Mol. Biol. Evol.*, **15**, 1447 (1998).
187 R. B. Russell and G. J. Barton, *Proteins*, **14**, 309 (1992).
188 W. R. Taylor, T. P. Flores, and C. A. Orengo, *Protein Sci.*, **3**, 1858 (1994).
189 W. R. Taylor, *J. Mol. Biol.*, **269**, 902 (1997).

Received for publication February 8, 1999.