BMC Bioinformatics

Research article

Improvement in accuracy of multiple sequence alignment using novel group-to-group sequence alignment algorithm with piecewise linear gap cost

Shinsuke Yamada*[1,2], Osamu Gotoh[2,3] and Hayato Yamana[1]

Address: [1]Department of Computer Science, Graduate School of Science and Engineering, Waseda University, 3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan, [2]Computational Biology Research Center (CBRC), National Institute of Advanced Industrial Science and Technology (AIST), 2-43 Aomi, Koto-ku, Tokyo 135-0064, Japan and [3]Department of Intelligence Science and Technology, Graduate School of Informatics, Kyoto University, Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan

Email: Shinsuke Yamada* - shinsuke@yama.info.waseda.ac.jp; Osamu Gotoh - gotoh@cbrc.jp; Hayato Yamana - yamana@yama.info.waseda.ac.jp

* Corresponding author

Published: 01 December 2006

BMC Bioinformatics 2006, **7**:524 doi:10.1186/1471-2105-7-524

Received: 20 June 2006
Accepted: 01 December 2006

This article is available from: http://www.biomedcentral.com/1471-2105/7/524

Abstract

Background: Multiple sequence alignment (MSA) is a useful tool in bioinformatics. Although many MSA algorithms have been developed, there is still room for improvement in accuracy and speed. In the alignment of a family of protein sequences, global MSA algorithms perform better than local ones in many cases, while local ones perform better than global ones when some sequences have long insertions or deletions (indels) relative to others. Many recent leading MSA algorithms have incorporated pairwise alignment information obtained from a mixture of sources into their scoring system to improve accuracy of alignment containing long indels.

Results: We propose a novel group-to-group sequence alignment algorithm that uses a piecewise linear gap cost. We developed a program called PRIME, which employs our proposed algorithm to optimize the well-defined sum-of-pairs score. PRIME stands for Profile-based Randomized Iteration MEthod. We evaluated PRIME and some recent MSA programs using BAliBASE version 3.0 and PREFAB version 4.0 benchmarks. The results of benchmark tests showed that PRIME can construct accurate alignments comparable to the most accurate programs currently available, including L-INS-i of MAFFT, ProbCons, and T-Coffee.

Conclusion: PRIME enables users to construct accurate alignments without having to employ pairwise alignment information. PRIME is available at http://prime.cbrc.jp/.

Background

Multiple sequence alignment (MSA) is a useful tool for elucidating the relationships among function, evolution, sequence, and structure of biological macromolecules such as genes and proteins [1-3]. Although we can calculate the optimal alignment of a set of sequences by *n*-dimensional dynamic programming (DP), the DP method is applicable to only a small number of sequences. In fact, even when a sum-of-pairs (SP) score with the simplest gap cost is used as an objective function, computation of optimal MSA is an NP-hard problem [4]. Hence, many heuristic methods have been developed. Almost all practical methods presently available adopt

either a progressive [5-7] or an iterative [8-10] heuristic strategy.

The group-to-group sequence alignment algorithm is a straightforward extension of the pairwise sequence alignment algorithm, and is the core of progressive and iterative methods. The essential difference of group-to-group sequence alignment from pairwise sequence alignment is the existence of gaps within each group of prealigned sequences. The gap opening penalty used in an affine gap cost disrupts the independence between adjacent columns, and hence calculating the optimal alignment between two groups with respect to the SP score was shown to be NP-complete [11]. Gotoh was the first to devise a group-to-group sequence alignment algorithm that optimizes the SP score by using a candidate list paradigm [12]. An algorithm with a candidate list paradigm, similar to the branch-and-bound method, prunes the candidates that are dispensable for arrival at an optimal solution. Kececioglu and Starrett proposed another candidate-pruning method [11]. Although these algorithms can calculate the optimal alignment between two groups, they require relatively extensive computational resources. Several papers have reported faster algorithms that use the heuristic estimation of gap opening penalties [8,10].

Several studies have discussed the tendency that global alignment methods perform better than local ones [13,14]. However, the opposite is also true when some sequences to be aligned have long insertions or deletions (indels). One reason for this tendency is that almost all group-to-group sequence alignment algorithms use an affine-like gap cost that over-penalizes long indels. To alleviate this problem, several methods have combined pairwise global and local alignments, or incorporated consistency information among pairwise alignments [6,15,16]. Another strategy to prevent over-penalizing long indels is to use a concave function as the gap cost. It is relatively easy to choose a concave gap cost that does not over-penalize long indels, and several pairwise sequence alignment algorithms using this gap cost have been developed [17,18]. However, there have been few attempts to incorporate this gap cost into a group-to-group sequence alignment algorithm to develop an MSA program.

In this paper, we propose a novel group-to-group sequence alignment algorithm with a piecewise linear gap cost [18], which is the key to a progressive or an iterative refinement method. The piecewise linear gap cost [18] is one of the concave functions and consists of L linear functions. Depending on the gap length, this gap cost varies its inclination, which corresponds to the gap extension penalty. However, in the case of group-to-group sequence alignment algorithm, it is difficult to calculate the proper gap extension penalty with only the data structures used in the previous algorithm that were designed to detect the opening of new gaps [12,19]. Accordingly, we newly introduce two additional data structures: 'insertion length profile' and 'dynamic gap information'. An insertion length profile vector is associated with each column of a group of sequences, while dynamic gap information keeps track of information about gaps inserted into a group during the DP process. Together with those used in the previous algorithm, gap extension penalty can be calculated efficiently. Using the proposed algorithm, we developed a program called PRIME.

PRIME stands for Profile-based Randomized Iteration MEthod. As a result of benchmark tests, the accuracy of our method is shown to be comparable to the most accurate methods available today, all of which incorporate pairwise alignment information obtained from all-by-all pairwise alignment. This implies that the piecewise linear gap cost is as effective as pairwise alignment information in improving the alignment accuracy of sequences, some of which have long indels.

Algorithms

In this section, we first review the previous group-to-group sequence alignment algorithm with an affine gap cost [12,19], and then describe a novel one with a piecewise linear gap cost. The final subsection outlines a doubly nested randomized iterative strategy with which our proposed algorithm is integrated.

The definitions of symbols are as follows. Let Σ be the residue set and $|\Sigma|$, the number of elements in Σ. Σ^* denotes the set containing a null and each element in Σ. A null means that a residue of one sequence does not aligned with that of another sequence when aligning sequences, and is denoted by the symbol '-'. A and B denote prealigned groups of sequences. A includes m rows, and B, n rows. The respective lengths of A and B are I and J. A_p, $\mathbf{a}_i$, and $a_{p,i}$ denote the p-th row of A, the i-th column of A and the i-th residue of A_p, respectively. B_q, $\mathbf{b}_j$, and $b_{q,j}$ are defined similarly. Both $a_{p,i}$ and $b_{q,j}$ belong to Σ^*. Note that any column of a group must not be a null column, which consists of nulls only. If all nulls are removed from a group, each row is an usual sequence. A run of consecutive nulls in a row is called a gap. A gap length is the number of nulls constituting the gap. A segment of A that consists of consecutive columns $\mathbf{a}_s$ to $\mathbf{a}_t$ is denoted by $A(s, t)$; A is also expressed as $A(1, I)$. $s(a, b)$ is a substitution score between residues a and b. By $g(x)$, we mean a gap cost function of gap length x. The pair weight between the p-th

sequence in A and the q-th sequence in B is $w_{p,q}$. If the three-way method [20] is used to calculate the pair weights, $w_{p,q}$ can be factorized as $w_{A_p} \cdot w_{B_q}$, where w_{A_p} and w_{B_q} are the weights for the p-th sequence in A and the q-th one in B, respectively.

Review of previous group-to-group sequence alignment algorithm with affine gap cost

The previous group-to-group sequence alignment algorithm that optimizes SP or weighted SP score with an affine gap cost is based on a two-dimensional DP method [12]. The key point of this algorithm is to exactly evaluate the gap opening penalties during the DP process. To explicitly consider gaps already present in each group, this algorithm introduces a gap state that denotes the number of consecutive nulls up to the current position.

Another important feature of this algorithm is the candidate list paradigm, which is a variant of branch-and-bound methods. Because the calculation of gap opening penalties depends on a previous partial DP path, simple extension of pairwise sequence alignment algorithm may not yield globally optimal alignment between two groups [12]. For rigorous calculation, not only locally optimal partial paths but also those that possibly contribute to globally optimal alignment have to be stored at each node of a DP matrix [11,12]. In the worst case, the number of candidates to be stored grows exponentially with the total number of sequences in the two groups [11]. As discussed in some papers [8,10,12], the group-to-group sequence alignment algorithm without the candidate list paradigm may suffice for good alignment. Moreover, because the novel group-to-group sequence alignment algorithm described below requires roughly twice as much computation time as the previous one at each DP process, we adopted a simpler algorithm without the candidate list paradigm.

Basic algorithm

Let an affine gap cost function be $g(x) = -(ux + v)$, where $u(> 0)$ and $v(> 0)$ are constants called gap extension penalty and gap opening penalty, respectively. The group-to-group sequence alignment algorithm with the affine gap cost employs essentially the same recurrent relations as the pairwise sequence alignment algorithm [21], with exact evaluation of gap opening and extension penalties. Like the pairwise sequence alignment algorithm, we calculate four variables at each node, (i, j), of the DP matrix: $H^0_{i,j}$, $H^1_{i,j}$, $H^2_{i,j}$, and $H^3_{i,j}$. $H^0_{i,j}$ holds the best score among $H^1_{i,j}$, $H^2_{i,j}$, and $H^3_{i,j}$ at (i, j). $H^1_{i,j}$ is a score of a partial alignment where $\mathbf{a}_i$ and $\mathbf{b}_j$ are aligned. $H^2_{i,j}$ and $H^3_{i,j}$ mean partial alignment scores where $\mathbf{a}_i$ and $\mathbf{b}_j$ are aligned with null columns, respectively. The recurrent equations are:

$$H^0_{i,j} = \max_{1\le k\le 3}\left\{H^k_{i,j}\right\} \quad (1)$$

$$H^1_{i,j} = H^0_{i-1,j-1} + G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1}) + S(\mathbf{a}_i, \mathbf{b}_j) \quad (2)$$

$$H^2_{i,j} = \max_{k=0,2}\left\{H^k_{i-1,j} + G(\mathbf{a}_i, -; P^k_{i-1,j})\right\} + S(\mathbf{a}_i, -) \quad (3)$$

$$H^3_{i,j} = \max_{k=0,3}\left\{H^k_{i,j-1} + G(-, \mathbf{b}_j; P^k_{i,j-1})\right\} + S(-, \mathbf{b}_j) \quad (4)$$

where '-' denotes a null column, and $P^0_{i-1,j-1}$ is a partial DP path. A partial DP path is one from (0, 0) to the node of the DP matrix in question, representing a partial alignment; for example, $P^0_{i,j}$ represents a partial alignment with the best score $H^0_{i,j}$ between $A(1, i)$ and $B(1, j) \cdot S(\mathbf{a}_i, \mathbf{b}_j)$ is responsible for the calculation of substitution scores and gap extension penalties:

$$S(\mathbf{a}_i, \mathbf{b}_j) = \sum_{1\le p\le m}\sum_{1\le q\le n} w_{p,q} \cdot s(a_{p,i}, b_{q,j}). \quad (5)$$

If either a or b is a null, $s(a, b) = -u$, and if both a and b are null, $s(a, b) = 0$. $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$ is the gap opening penalty when $\mathbf{a}_i$ is aligned with $\mathbf{b}_j$:

$$G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1}) = \sum_{1\le p\le m}\sum_{1\le q\le n} w_{p,q} \cdot (-v) \cdot \gamma(a_{p,i}, b_{q,j}, x^0_{p,i-1}, \gamma^0_{q,j-1}). \quad (6)$$

$x^0_{p,i-1}$ and $\gamma^0_{q,j-1}$ are the gap states for the p-th and q-th rows in $A(1, i - 1)$ and $B(1, j - 1)$ on $P^0_{i-1,j-1}$, respectively. As mentioned above, the gap state is the length of the gap up to the current position. $\gamma(a, b, x, \gamma)$ represents whether a gap opens with respect to a pair of rows. Specifically, if a is a residue, $x \ge \gamma$, and b is a null; or if a is a null, $x \le \gamma$, and b is a residue; then $\gamma(a, b, x, \gamma) = 1$. Otherwise, $\gamma(a, b, x, \gamma) = 0$. In order to evaluate exact gap openings for calculation of each $H^k_{i,j}$, the gap states must be updated. If $a_{p,i}$ is a null, $x^0_{p,i} = x^d_{p,i-1} + 1$ where $d = \arg\max_{1\le k\le 3}\{H^k_{i,j}\}$.

Otherwise, $x^0_{p,i} = 0$. The other gap states are calculated in a similar way.

Use of generalized profile

Although equations 5 and 6 require $O(mn)$ computational steps, these steps can be reduced by using a generalized profile [19] and the three-way weighting method [20]. The idea of using the generalized profile is that the same residue types or gap states on a column are treated together. The generalized profile consists of four vectors calculated from each column of a group: frequency, residue profile, and two kinds of static gap profile vectors. These vectors can be obtained in advance of the DP process. The frequency and residue profile vectors are used to calculate $S(\mathbf{a}_i, \mathbf{b}_j)$, while the static gap profile vectors are necessary to calculate $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$.

With the frequency and residue profile vectors, $S(\mathbf{a}_i, \mathbf{b}_j)$ is calculated by

$$S(\mathbf{a}_i, \mathbf{b}_j) = \sum_{r \in \Sigma^*} f_{\mathbf{a}_i,r} \cdot p_{\mathbf{b}_j,r} = \sum_{r \in \Sigma^*} p_{\mathbf{a}_i,r} \cdot f_{\mathbf{b}_j,r} \qquad (7)$$

where residue frequency $f_{\mathbf{a}_i,r}$ is the weighted frequency of occurrence of residue type r (including null) on column $\mathbf{a}_i$, and residue profile $p_{\mathbf{b}_j,r} = \sum_{t \in \Sigma^*} f_{\mathbf{b}_j,r} \cdot s(r,t)$. Both $p_{\mathbf{a}_i,r}$ and $f_{\mathbf{b}_j,r}$ are defined in the same way. The right hand side of equation 7 requires $O(|\Sigma^*|)$, because each frequency and residue profile vector consists of $|\Sigma^*|$ values. When $m \times n$ is sufficiently large, the computation time can be considerably reduced. Although $S(\mathbf{a}_i, \mathbf{b}_j)$ is easy to calculate, the profile-based calculation of $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$ is somewhat complicated, because, in addition to static gaps, dynamic gaps must be considered explicitly. A static gap consists of consecutive static nulls that already exist in each group, while a dynamic gap denotes a run of dynamic nulls that are inserted into each group during the DP process. Note that we distinguish static and dynamic gaps for convenience of the description of our algorithm, while they contribute to the total alignment score in the same way. Let us consider an example of the gap opening penalty calculation when $\mathbf{a}_8$ is aligned with $\mathbf{b}_{12}$ (Figure 1). To keep the discussion simple, we consider A_1 and B_2 only. From simple observation, we find that a gap between A_1 and B_2 has already opened before $\mathbf{a}_8$ is aligned with $\mathbf{b}_{12}$. However, if the gap opening penalty were calculated using the static gap states only, a gap opening would be detected wrongly, because $a_{1,8}$ is a null, $b_{2,12}$ is a residue, and $x_{1,7} = 0 \le y_{2,11} = 0$ where $x_{1,7}$ and $y_{2,11}$ are the static gap states of $a_{1,7}$ and $b_{2,11}$, respectively. For correct detection of gap opening, we need 'running gap states', each of which represents the sum of the numbers of consecutive static and dynamic nulls up to the current position. For the example shown in Figure 1, the running gap state for A_2 at column position $\mathbf{a}_7$ is 11, which is composed of 7 static nulls and 4 dynamic nulls.

Static gap states are compactly represented as a static gap profile. A static gap profile is obtained by gathering static gap states with the same values at a column. More specifically, the static gap profile at a column $\mathbf{c}_k$ consists of two vectors $E_{\mathbf{c}_k}$ and $S_{\mathbf{c}_k}$. Each element of both vectors has the same form: $\{(g, f)\}$ where g is the gap state of previous column $\mathbf{c}_{k-1}$ and f is the weighted frequency of the occurrence of rows whose element on $\mathbf{c}_k$ is either a residue or null depending on $E_{\mathbf{c}_k}$ or $S_{\mathbf{c}_k}$, respectively. Although $E_{\mathbf{c}_k}$ itself may be used to calculate gap opening penalties, its accumulated form, $E^+_{\mathbf{c}_k}$, is more convenient to reduce the computation time. If an element in $E^+_{\mathbf{c}_k}$ is (g, f^+), f^+ represents the weighted frequency of the occurrence of rows whose gap states are not less than g. For the example shown in Figure 1, $E_{\mathbf{a}_8}$, $E^+_{\mathbf{a}_8}$, and $S_{\mathbf{a}_8}$ are $\{(0, w_{A_4}), (2, w_{A_3}), (7, w_{A_2})\}$, $\{(0, w_{A_4} + w_{A_3} + w_{A_2}), (2, w_{A_3} + w_{A_2}), (7, w_{A_2})\}$, and $\{(0, w_{A_1})\}$, respectively. Since all gap states are different in the worst case, the total number of elements of $E^+_{\mathbf{c}_k}$ and $S_{\mathbf{c}_k}$ is at most the number of sequences in the group.

Like a static gap profile, a running gap profile can represent running gap states compactly. In order to obtain a running gap profile, another data structure called a gap mediation profile is required. A gap mediation profile is defined for each path in the DP process, and records the total number of dynamic nulls inserted into static gaps that are currently open. Each element of the gap mediation profile is expressed as (s, d), where s is the length of a static gap and d is the summed length of dynamic gaps inserted within or after the static gap. Let $M^0_{i,j}(A)$ be the

i		1	2			3	4	5	6	7		8
A_1	–	·	·	–	–	*	*	·	*	*	–	·
A_2	–	·	·	–	–	·	·	·	·	·	–	*
A_3	–	*	*	–	–	*	*	*	·	·	–	*
A_4	–	*	*	–	–	*	*	*	*	*	–	*
B_1	*	*	*	*	·	·	·	·	·	·	·	·
B_2	*	*	*	*	*	*	*	*	*	*	*	*
j	1	2	3	4	5	6	7	8	9	10	11	12

Figure 1

Example of gap extension penalty calculation. This figure shows an example of columns $\mathbf{a}_8$ and $\mathbf{b}_{12}$ being aligned. '*', '·', and '-' denote a residue, a static null, and a dynamic null, respectively. We assume that piecewise linear gap cost $g(x)$ is $\max_{k=1,2}\{-(u_k x + v_k)\}$ and critical gap length x_c (= $\lfloor (v_2 - v_1)/(u_1 - u_2) \rfloor$) is 4. Gap extension penalty is u_1 if $x \leq 4$, otherwise u_2. Running gap profile vectors $\hat{S}_{\mathbf{a}_8}$ and $\hat{E}^+_{\mathbf{a}_8}$ are {(1, w_{A_1})} and {(1, $w_{A_2} + w_{A_3} + w_{A_4}$), (3, $w_{A_2} + w_{A_3}$), (11, w_{A_2})}, respectively. Dynamic gap information $D^0_{7,11}(A)$ is {(0, 1), (2, 2), (7, 1)}. Segment profile $F_{\mathbf{a}_8}$ is {(1, w_{A_2}), (3, $w_{A_2} + w_{A_3}$), (5, $w_{A_2} + w_{A_3} + w_{A_4}$)}. Similarly, the profile vectors of B are defined: $\hat{S}_{\mathbf{b}_{12}}$ = {(7, w_{B_1})}, $\hat{E}^+_{\mathbf{b}_{12}}$ = {(0, w_{B_2})}, $D^0_{7,11}(B)$ is empty, and $F_{\mathbf{b}_{12}}$ = {(1, 0), (9, w_{B_2})}. In what follows, we consider the non-trivial calculation of the gap extension penalty with respect to the gap of B_1, the target gap. By using $\hat{S}_{\mathbf{b}_{11}}$ and $D^0_{7,11}(A)$, we find that the two dynamic gaps specified by (2, 2) and (7, 1) in $D^0_{7,11}(A)$ are partially and completely aligned with the target gap, respectively. Consequently, the total number of nulls aligned with null columns of dynamic gaps to be removed is 2. Therefore, the number of columns of B_1 is 5(= 7 - 2). By subtracting 5 from 8 (the end position of the segment), the starting position of A, 3, is obtained. Then, the gap extension penalty with respect to the gap of B_1 is w_{B_1} ($F_1 \cdot u_1 + F_2 \cdot u_2$) where F_1 = $w_{A_2} + w_{A_3}$ and F_2 = w_{A_4}. Note that A_1 is not involved in the gap extension penalty because $a_{1,8}$ is a null.

gap mediation profile of group A at the DP node $(i, j) \cdot M^0_{i,j}(A)$ follows a recurrent relation, the initial condition of which is $M^0_{i,j}(A) = \{(0, 0)\}$. We first consider the case where $\mathbf{a}_i$ and $\mathbf{b}_j$ are aligned. For each (g, f) in $S_{\mathbf{a}_i}$, if there exists element (s, d) in $M^0_{i-1,j-1}(A)$ such that $g = s$, then $(s + 1, d) \in M^0_{i,j}(A)$. In the case where $\mathbf{b}_j$ is aligned with a null column, then $(s, d + 1) \in M^0_{i,j}(A)$ where (s, d) is an element in $M^0_{i,j-1}(A)$ or $M^3_{i,j-1}(A)$ depending on the maximum operation of equation 4. If $\mathbf{a}_i$ is aligned with a null column, $M^0_{i,j}(A)$ equals $M^0_{i-1,j}(A)$ or $M^2_{i-1,j}(A)$. In the case of Figure 1, $M^0_{7,11}(A) = \{(0, 1), (2, 1), (7, 4)\}$ is derived from $M^0_{7,10}(A) = \{(0, 0), (2, 0), (7, 3)\}$. The other gap mediation profile vectors are constructed in a similar way.

Running gap profile vectors $\hat{E}^+_{\mathbf{a}_i}$ and $\hat{S}_{\mathbf{a}_i}$ are obtained by combining gap mediation profile $M^0_{i-1,j-1}(A)$, and static gap profiles $E^+_{\mathbf{a}_i}$ and $S_{\mathbf{a}_i}$, respectively. For each (g, f^+) in $E^+_{\mathbf{a}_i}$, (s, d) is chosen from $M^0_{i-1,j-1}(A)$ such that $g = s$, and then $(s + d, f^+) \in \hat{E}^+_{\mathbf{a}_i}$. Similarly, $\hat{S}_{\mathbf{a}_i}$ is obtained from $M^0_{i-1,j-1}(A)$ and $S_{\mathbf{a}_i}$. For example, $\hat{E}^+_{\mathbf{a}_8}$ and $\hat{S}_{\mathbf{a}_8}$ at the node (8, 12) of Figure 1 are {(1, $w_{A_2} + w_{A_3} + w_{A_4}$), (3, $w_{A_2} + w_{A_3}$), (11, w_{A_2})} and {(1, w_{A_1})}, respectively.

Using running gap profile vectors $\hat{E}^+_{\mathbf{a}_i}$, $\hat{S}_{\mathbf{a}_i}$, $\hat{E}^+_{\mathbf{b}_j}$, and $\hat{S}_{\mathbf{b}_j}$, Equation 6 can be rewritten as:

$$G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1}) = (-v) \cdot (S_{\mathbf{b}_j} \bullet E^+_{\mathbf{a}_i} + S_{\mathbf{a}_i} \bullet E^+_{\mathbf{b}_j}). \quad (8)$$

$S \bullet E^+$ is expressed as

$$S \bullet E^+ = \sum_{1 \leq p \leq |S|} f(s_p) \cdot f(e^+_q)$$

where s_p and e^+_q are the p-th and q-th elements in S and E^+, respectively, and $f(e)$ is the weighted frequency of element e. q is chosen such that the gap state of e^+_q is the smallest among the elements in E^+ whose gap states are not less than that of s_k. Calculation of $S \bullet E^+$ requires $O(|S|) + |E^+|)$, and hence $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$ at most $O(m + n)$. Using equation 8 instead of equation 6 can reduce computation from

$O(mn)$ to $O(m + n)$ even in the worst case where gap states on a column are mutually different.

Novel group-to-group sequence alignment algorithm with piecewise linear gap cost

In this section, we describe a novel group-to-group sequence alignment algorithm with a piecewise linear gap cost. Although this algorithm uses recurrent equations 1 to 4, the algorithms calculating $S(\mathbf{a}_i, \mathbf{b}_j)$ and $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$ must be changed. Roughly speaking, the term for calculating gap extension penalties is transferred from $S(\mathbf{a}_i, \mathbf{b}_j)$ to $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$. After explanation of the piecewise linear gap cost, we describe these algorithms in detail.

Piecewise linear gap cost

The piecewise linear gap cost consists of several linear functions [18]:

$$g(x) = \max_{1 \le l \le L} \{-(u_l x + v_l)\} \qquad (9)$$

where $u_l > u_{l+1} (\ge 0)$ and $v_{l+1} > v_l (> 0)$. When $L = 1$, this cost is the same as the affine gap cost. This cost could alleviate over-penalizing long indels, because the inclination of $g(x)$, which corresponds to a gap extension penalty, u_l, becomes small as gap length increases. In other words, this cost calculates gap extension penalties based on gap length. For the sake of simplicity, we restricted our attention to the case of $L = 2$. Then, $g(x) = -(u_1 x + v_1)$ if $x \le x_c$ or $g(x) = -(u_2 x + v_2)$, otherwise, $x_c = \lfloor (v_2 - v_1)/(u_1 - u_2) \rfloor$ is called the critical gap length.

Calculation of substitution score

As mentioned above, the calculation of gap extension penalties must be separated from the calculation of the substitution score $S(\mathbf{a}_i, \mathbf{b}_j)$ in order to use the piecewise linear gap cost. Therefore, $S(\mathbf{a}_i, \mathbf{b}_j)$ is expressed as:

$$S(\mathbf{a}_i, \mathbf{b}_j) = \sum_{r \in \Sigma} f_{\mathbf{a}_i, r} \cdot p_{\mathbf{b}_j, r} = \sum_{r \in \Sigma} p_{\mathbf{a}_i, r} \cdot f_{\mathbf{b}_j, r}. \qquad (10)$$

where $p_{\mathbf{b}_j, r} = \sum_{t \in \Sigma} f_{\mathbf{b}_j, r} \cdot s(r, t)$. Note that this equation and the definition of residue profile vector sum over not Σ^* but Σ.

Calculation of gap extension penalty

In the previous algorithm with an affine gap cost, $G(\mathbf{a}_i, \mathbf{b}_j; P^0_{i-1,j-1})$ was responsible for the gap opening penalty only; however, it must take care of the gap extension penalty in the case of the piecewise linear gap cost. Therefore, a term for gap extension penalty is added to the right hand side of equation 8. Let us consider an example of gap extension penalty calculation for the null on $\mathbf{b}_{12}$ aligned with the residues on $\mathbf{a}_8$ (the last column of Figure 1). The gap state at $b_{1,12}$ is 8. With omission of nulls that are aligned with other nulls, the lengths of the gap on B_1 relative to A_2, A_3, and A_4 are 1, 4, and 6, respectively. Assuming that the critical gap length $x_c = 4$, we obtain the respective gap extension penalties for A_2, A_3, and A_4 as u_1, u_1, and u_2. Therefore, the gap extension penalty in question is $w_{A_1} (F_1 \cdot u_1 + F_2 \cdot u_2)$, where $F_1 = w_{A_2} + w_{A_3}$ and $F_2 = w_{A_4}$.

This example indicates two important points for the exact calculation of gap extension penalty. First, each gap length is obtained by counting the residues on each row of a specific range in A called 'relevant segment', $A(3, 8)$ in this example. Second, the two nulls on B_1 at columns $\mathbf{b}_5$ and $\mathbf{b}_{11}$ are aligned with two separate dynamic gaps inserted into A. The first observation suggests that F_1 and F_2 for any segment may be calculated before the DP process. However, the second observation indicates that the number of null columns of dynamic gaps aligned with static nulls on a target gap must be subtracted from the gap state of the target gap for correct assignment of the relevant segment; without this subtraction, the relevant segment would be assigned as $A(1, 8)$ in the above example.

We initially consider the first point, neglecting the presence of dynamic gaps for a moment. Without loss of generality, we assume that the residues on $\mathbf{a}_i$ are aligned with a null of a target gap on $\mathbf{b}_j$ whose gap state is g. We define $F_0(\mathbf{a}_i)$ as the weighted fraction of nulls on $\mathbf{a}_i$: $F_0(\mathbf{a}_i) \equiv \sum_{p=1}^{m} w_{A_p} \delta(a_{p,i}, -)$, where $\delta(a, b) = 1$ if $a = b$, otherwise, $\delta(a, b) = 0$, and '-' denotes a null. $F_0(\mathbf{a}_i)$ is the same as the null component of the frequency profile at $\mathbf{a}_i$, *i.e.* $F_0(\mathbf{a}_i) = f_{\mathbf{a}_i, -}$. We also define $F_1(A, i, g)$ as the sum of weights of sequences $\{A_p\}$ such that $a_{p,i} \ne$ '-' and the number of residues within segment $A(i - g + 1, i)$ aligned with the target gap is less than or equal to x_c: $F_1(A, i, g) = \sum_{p=1}^{m} w_{A_p} (1 - \delta(a_{p,i}, -)) \Delta(a_p, i, g)$, where $\Delta(a_p, i, g) = 1$ if the number of residues on the p-th row of $A(i - g + 1, i)$ is less than or equal to x_c, otherwise, $\Delta(a_p, i, g) = 0$. Specifically, if $\sum_{k=i}^{i-g+1} (1 - \delta(a_{p,k}, -)) \le x_c$, then

$\Delta(a_p, i, g) = 1$, otherwise, $\Delta(a_p, i, g) = 0$. Likewise, $F_2(A, i, g)$ is defined as the sum of weights of sequences $\{A_p\}$ such that $a_{p,i} \neq$ '-' and the number of residues within $A(i - g + 1, i)$ aligned with the target gap is greater than x_c. Obviously,

$$F_0(\mathbf{a}_i) + F_1(A, i, g) + F_2(A, i, g) = 1. \quad (11)$$

Because there may exist gap states g and g' $(0 \leq g < g' \leq i)$ such that $F_1(A, i, g) = F_1(A, i, g')$, we need to store only distinct $F_1(A, i, g)$ values, the number of which is at most the number of rows of A, m. For actual calculations, we use an 'insertion length profile' associated with each column of a group. An insertion length profile of column $\mathbf{a}_i$, $F_{\mathbf{a}_i}$ is expressed as $F_{\mathbf{a}_i} = \{(i - g_k + 1, F_1(A, i, g_k))\}$, where g_k is a maximum gap state such that $F_1(A, i, g_k) = F_1(A, i, g')$ for $g_k \leq g' < g_{k+1}$. Information about $F_2(A, i, g)$ does not need to be recorded, because it can be easily derived from $F_0(\mathbf{a}_i)$ and $F_{\mathbf{a}_i}$ through equation 11.

The second point raises the key problem: how to determine the 'tailored gap state' of the target gap, $\hat{g} = g - s$, where g is the gap state of the target gap and s is the number of dynamic null columns to be removed for correct assignment of the relevant segment. For example, dynamic null columns aligned with $\mathbf{b}_5$ and $\mathbf{b}_{11}$ in Figure 1 are removed. To keep track of the dynamic gaps to be removed, we newly introduce the 'dynamic gap information' list. Each element of dynamic gap information is represented by (p, l), where p and l indicate the position and the length of a dynamic gap, respectively. For efficiency, the dynamic gap information list $\{(p_k, l_k)\}$ is sorted in order of p_k. The dynamic gap information of group A at (i, j), $D^0_{i,j}(A)$, is recurrently calculated as follows. When $\mathbf{a}_i$ and $\mathbf{b}_j$ are aligned, $D^0_{i,j}(A)$ is simply copied from $D^0_{i-1,j-1}(A)$. Similarly, when $\mathbf{a}_i$ is aligned with a null column, $D^0_{i,j}(A)$ is copied from $D^0_{i-1,j}(A)$ or $D^2_{i-1,j}(A)$ depending on the maximum operation of recurrent equation 3. When $\mathbf{b}_j$ is aligned with a null column, $D^0_{i,j}(A)$ is first copied from $D^0_{i,j-1}(A)$ or $D^3_{i,j-1}(A)$, and then a new element is added to it or its last element is modified. Specifically, if the last element of $D^0_{i,j}(A)$ is (i, l), it is modified to $(i, l + 1)$. Otherwise, a new element $(i, 1)$ is added to $D^0_{i,j}(A)$. The other dynamic gap information lists are obtained in a similar way.

It is worth mentioning that dynamic gap information $D^0_{i,j}(A)$ and gap mediation profile $M^0_{i,j}(A)$ contain information on dynamic gaps from different viewpoints. The information held in dynamic gap information is relevant to the group aligned with a gap in question (group A in the present example), while that maintained in a gap mediation profile is relevant to the group containing the gap (group B in the above example). In addition, each element of $D^0_{i,j}(A)$ refers to a single dynamic gap, while that of $M^0_{i,j}(A)$ records the total length of separate dynamic gaps inserted within or after a static gap. In the case of Figure 1, each element of $D^0_{8,12}(A) = \{(0, 1), (2, 2), (7, 1)\}$ indicates the dynamic gaps of lengths 1, 2, and 1 inserted before column $\mathbf{a}_1$, after $\mathbf{a}_2$, and after $\mathbf{a}_7$, respectively, while that of $M^0_{7,11}(A) = \{(0, 1), (2, 1), (7, 4)\}$ means the total length of dynamic gaps inserted after the last non-null residue within or before the present column in A_1 or A_4, A_3, and A_2, respectively.

Given a dynamic gap information list $D_{i,j}(A) = \{(p_k, l_k)\}$ and a gap state g, we can easily derive the tailored gap state $\hat{g}$ with the following algorithm, in which s denotes the total number of dynamic gap columns to be removed and t_k indicates the distance in the reverse direction from the current position i to the start position of the k-th dynamic gap inserted into $A(1, i)$:

Algorithm getTailoredGapState($D_{i,j}(A)$, g)

1. $s \leftarrow 0$

2. for $k \leftarrow |D_{i,j}(A)|$ down to 1

(a) $t_k \leftarrow s + i - p_k + l_k$

(b) if $t_k \leq g$, then $s \leftarrow s + l_k$

(c) else if $t_k - l_k < g$, then $s \leftarrow s + g - (t_k - l_k)$

(d) if $t_k \geq g$, then return $\hat{g} \leftarrow g - s$

3. retrun $\hat{g} \leftarrow g - s$

The gap extension penalty for the target gap is then obtained by

$$w_B\{u_1 \cdot F_1(A, i, \hat{g}) + u_2 \cdot F_2(A, i, \hat{g})\}, \quad (12)$$

where w_B is the weight given to the sequence containing the target gap. Note that step 2c in this algorithm examines whether or not the dynamic gap in question is partially aligned with the target gap. For the example considered in Figure 1, $t_3 = 0 + 8 - 7 + 1 = 2$, and $s = l_3 = 1$ are calculated after the first iteration of steps 2 in this algorithm. In the second iteration, we obtain $t_2 = 1 + 8 - 2 + 2 = 9$. Because $t_2 = 9 > g = 8$ and $t_2 - l_2 = 7 < g = 8$, we can recognize that the dynamic gap is partially aligned with the target gap, and hence the number of dynamic null columns to be removed 8 - 7 = 1 is added to s. The gap extension penalty, u, summed over all the nulls on column $\mathbf{b}_j$ can easily be calculated with the following algorithm:

1. $u \leftarrow 0$

2. for $h \leftarrow 1$ to $|\hat{S}_{\mathbf{b}_j}|$

(a) $\hat{g} \leftarrow$ getTailoredGapState($D_{i,j}(A)$, g_h)

(b) $u \leftarrow u + f_h\{u_1 \cdot F_1(A, i, \widehat{g_h}) + u_2 \cdot F_2(A, i, \widehat{g_h})\}$

3. return u

where (g_h, f_h) is the h-th element of $\hat{S}_{\mathbf{b}_j}$. Because we have already prepared running gap state profile $\hat{S}_{\mathbf{b}_j}$, the computation is done in $O(|D_{i,j}(A)| \cdot |\hat{S}_{\mathbf{b}_j}|)$. To obtain the total gap extension penalty at each DP step, we must also consider the opposite situation where residues on $\mathbf{b}_j$ are aligned with gaps on $\mathbf{a}_i$ in a similar way.

Doubly nested randomized iterative strategy

The doubly nested randomized iterative strategy involves refinement of alignment, phylogenetic tree, and pair weights until these are mutually consistent [9]. After preparation of an initial alignment with such progressive methods as the oligomer counting based method [8,10], this strategy refines the initial alignment as follows:

1. Calculate a distance matrix from the multiple alignment

2. Construct a phylogenetic tree from the distance matrix

3. Calculate pair weights from the phylogenetic tree

4. Iteratively refine the alignment using the phylogenetic tree and the pair weights

(a) Divide the alignment into two groups based on a randomly chosen branch of the tree

(b) Align these two groups using a group-to-group sequence alignment algorithm

(c) Repeat steps 4a to 4b until no better weighted SP score is obtained

5. Repeat steps 1 to 4 until the weighted SP score of the alignment does not improve anymore at step 4

Results

PRIME

We developed a program called PRIME (Profile-based Randomized Iteration Method). PRIME is written in ISO standard C++, implementing the doubly nested randomized iterative strategy similar to our previous MSA program, Prrn [9]. However, PRIME employs our proposed algorithm with a piecewise linear gap cost in contrast to Prrn that uses an affine gap cost. Another algorithmic difference between PRIME and Prrn is that the latter uses the candidate list paradigm in the group-to-group sequence alignment algorithm and the anchoring method, whereas the former adopts a simpler DP method without anchoring heuristics. The parameters of PRIME including selection of substitution matrix and gap cost parameters are optimized using an older BAliBASE, version 2.01 [22]. Because only about 20% of the sequences in BAliBASE version 3.0 [23] used for the test are common to those in BAliBASE version 2.01, we do not think that these parameters are over-fitted against BAliBASE version 3.0. Initial MSAs are constructed using a simple progressive method with the proposed group-to-group sequence alignment algorithm based on a distance matrix calculated from pairwise sequence alignment. The PRIME source code is provided as an additional file [see Additional file 1], and can be downloaded at PRIME website [24]. The future version of PRIME will be available at this site.

Benchmarks

To evaluate the performance of PRIME and other MSA programs shown in Table 1, we execute two benchmarks: BAliBASE version 3.0 [22,23,25] and PREFAB version 4.0 [8]. In the case of PREFAB, we also test two global pairwise sequence alignment programs as controls: $PSA_{piecewise}$ and PSA_{affine}. $PSA_{piecewise}$ and PSA_{affine} use the piecewise linear gap cost and the affine gap cost, respectively.

BAliBASE

BAliBASE is categorized into five references according to the nature of sequences to be aligned (Table 2). Reference

Table 1: List of evaluated programs

program	version	option
$PRIME_{piecewise}$		blosum62, $g(x) = \max\{-(x + 9), -(0.5x + 21.5)\}$
$PRIME_{affine}$		blosum62, $g(x) = -(x + 9)$
Prrn [9]	3.4	-b2 -mblosum62 -u1 -v9
MAFFT* [15]	5.662	--maxiterate 1000 --localpair (L-INS-i)
ProbCons* [16]	1.09	default
T-Coffee* [27]	2.02	default
MUSCLE [8]	3.52	default
DIALIGN-T [28]	0.2.1	default
POA [5]	2	-do_global -do_progressive blosum80_trunc.mat
ClustalW [7]	1.83	default

Programs with * employ pairwise alignment information when calculating multiple alignment. Parameters or options of each program other than the gap penalty ones are chosen to obtain as accurate an alignment as possible.

1 is further divided into two sub-references based on sequence identities. Although the previous BAliBASE version 2.01 has been widely used, it had a problem that some sequences were trimmed off non-homologous regions [26]. Therefore, two test sets are prepared in BAliBASE version 3.0: full length set and homologous region set. Each sequence in the full length set is not trimmed off non-homologous regions, whereas the homologous region set consists of alignments of trimmed sequences and hence corresponds to the previous BAliBASE. However, reference 4 is excluded from the homologous region set due to its objective.

Alignment evaluation based on BAliBASE

To evaluate alignment accuracy based on BAliBASE, we use sum-of-pairs and column scores [14]. The sum-of-pairs score (*SPS*) is defined as the proportion of correctly aligned residue pairs:

$$SPS = \frac{\sum_{i=1}^{I} SP_i^t}{\sum_{j=1}^{J} SP_j^r},$$

where I and J are the number of columns of test and reference alignments, respectively. SP_i^t is defined as:

$$SP_i^t = \sum_{1 \le m < n \le N} p_i(m,n).$$

If aligned residue pair a_{mi} and a_{ni} of the test alignment also exists in the reference alignment, $P_i(m, n) = 1$. Otherwise, $p_i(m, n) = 0$. SP_j^r is the total number of aligned pairs on column j of the reference alignment. The column score (*CS*) represents the proportion of correctly aligned columns:

$$CS = \frac{1}{J}\sum_{i=1}^{I} c_i.$$

If the column of the test alignment is identical to the i-th column of the reference alignment, $c_i = 1$.

Otherwise, $c_i = 0$.

PREFAB

PREFAB is another MSA benchmark. Each alignment of PREFAB is generated automatically, while that of BAliBASE is constructed by human expertise. PREFAB consists of three data sets: main, long gap, and weighting sets. The main set corresponds to the previous PREFAB version 3.0, which is not categorized unlike BAliBASE. Each alignment of the long gap set, a subset of the main set, contains one or more gaps whose lengths are more than 10. The weighting set involves alignments each of which includes more sequences of one sub-family than that of the other sub-families. Whereas each reference alignment of BAliBASE is provided as an MSA, each reference alignment of PREFAB

Table 2: BAliBASE version 3.0 contents

	no. of alignments	characteristic of alignment
Reference 1.1	37	phylogenetically equidistant (less than 20% identity)
Reference 1.2	42	phylogenetically equidistant (20 to 40% identity)
Reference 2	39	families including orphan sequences
Reference 3	29	equidistant families (less than 25% identity)
Reference 4	48	long N/C terminal extensions (excluded from homologous region set)
Reference 5	14	long internal insertions

is provided as a pairwise alignment of a pair of PDB sequences of known structures.

Alignment evaluation based on PREFAB

For alignment evaluation of PREFAB, the quality score is employed, which measures only two PDB sequences within each alignment. The quality score (*QS*) is the ratio of correctly aligned residue pairs of the reference pairwise alignment: $QS = \frac{1}{J}\sum_{i=1}^{J} p_i$, where *J* is the number of residue pairs in the reference alignment. If the residue pair of the test alignment is also aligned in the reference alignment, $p_i = 1$. Otherwise, $p_i = 0$. Note that if the reference alignment is pairwise alignment, quality score, sum-of-pairs score, and column score have the same value.

Results of BAliBASE benchmark test

Full length set

The average sum-of-pairs and column scores of the full length set are shown in Tables 3 and 4, respectively. The last columns of both tables are the rank sum of the Friedman test. The program with the smallest rank sum means that the program consistently constructs the most accurate alignments even if it does not achieve the largest average score. The *p*-values of the Friedman test are shown in the additional file [see Additional file 2]. The Friedman test indicates that the tested programs are classified into three groups according to their performances. The most accurate group consists of PRIME$_{piecewise}$, PRIME$_{affine}$, MAFFT, ProbCons, and T-Coffee. The second most accurate one is Prrn and MUSCLE. The accuracies of DIALIGN-T, POA, and ClustalW are comparable to each other but are significantly lower than those of Prrn and MUSCLE. Figure 2 shows the performance difference between PRIME$_{piecewise}$ and PRIME$_{affine}$. In this figure, we plot the difference in alignment scores of PRIME$_{piecewise}$ and PRIME$_{affine}$. A positive difference score means that PRIME$_{piecewise}$ constructs more accurate alignments than PRIME$_{affine}$, and *vice versa*. Although the difference is not statistically significant, PRIME$_{piecewise}$ shows better performance than PRIME$_{affine}$, as expected.

Homologous region set

The average sum-of-pairs and column scores of the homologous region set are shown in Tables 5 and 6, respectively. The *p*-values of the Friedman test are shown in the additional file [see Additional file 3]. Unlike the results of the full length set, little difference in accuracy was detected between PRIME$_{piecewise}$ and PRIME$_{affine}$. Figure 3 shows the performance difference between them. Because the terminal non-homologous regions trimmed off in the homologous region set are usually long, for the full length set, the piecewise linear gap cost treats the long terminal gaps more effectively than the affine gap cost, and hence PRIME$_{piecewise}$ shows better performance than PRIME$_{affine}$. However, because the difference between these gap costs is relatively small in the homologous region set, PRIME$_{piecewise}$ and PRIME$_{affine}$ show similar performance. The relative performance of the nine programs examined is nearly the same as that of the full length set in a statistical sense.

Effects of non-homologous regions

We examine the effects of non-homologous regions in more detail. The critical difference between the full length and homologous region sets is the existence of non-homologous regions at N/C terminals. Therefore, the difference in alignment scores obtained by the same program for the corresponding members in the full length and homologous region sets indicates to what extent the program properly deals with terminal gaps. Figures 4 and 5 show the average difference of sum-of-pairs and column scores between the full length and homologous region sets. Each difference score is calculated by subtracting the alignment score of the full length set from that of the homologous region set. A positive difference score means that the non-homologous regions adversely affect alignment accuracy, whereas a negative score indicates improvement in alignment accuracy due to the presence of such regions. If the score difference is close to 0, the program is considered to be robust against the non-homologous regions. The results indicate that PRIME$_{piecewise}$ is less affected by such regions than PRIME$_{affine}$. This follows the general tendency that terminal gaps reduce more significantly the accuracy of global alignment programs including Prrn, MUSCLE, POA, and ClustalW than that of MAFFT, ProbCons, and T-Coffee that incorporate local alignment information in some ways. These observations indicate that PRlME$_{piecewise}$ deals with terminal gaps better than conventional global MSA programs, although not as well as those incorporating local alignment information.

Results of PREFAB benchmark test

The average quality scores of the three sets of PREFAB are shown in Table 7. The overall tendencies of relative performances and the Friedman tests of the main and long gap sets are nearly the same as those of BAliBASE. However, in the case of the weighting set, all programs except POA are comparable to each other. Because POA does not use sequence weights, 'biased sub-family composition' might adversely affect the performance of POA compared with the other programs.

Computation time

The computation time of each program for executing the benchmarks is compiled in Table 8. The computer we used is Pentium3 933 MHz with 1 GB memory, running on RedHat Linux 7.3. PRIME$_{piecewise}$ and PRIME$_{affine}$ are

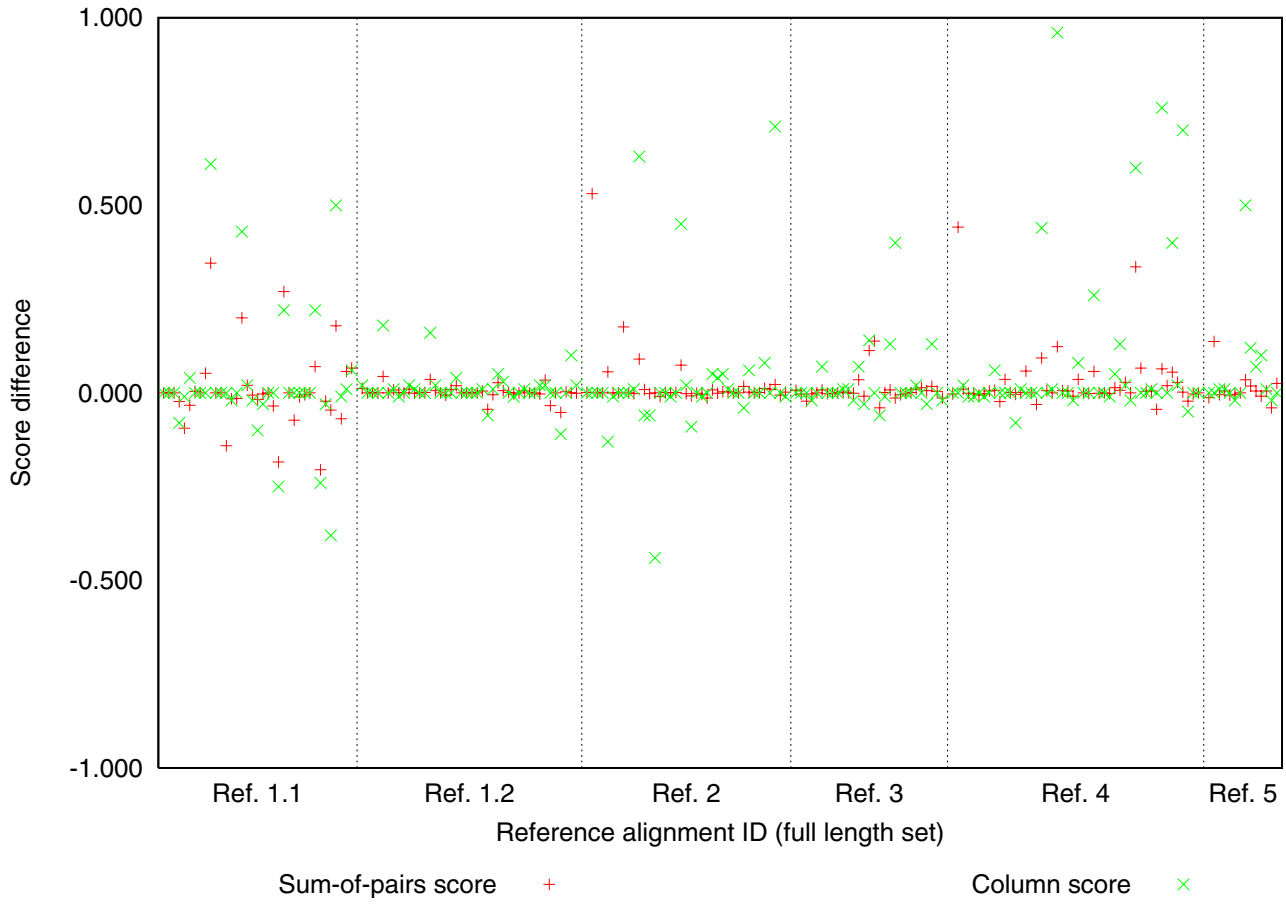

Figure 2
Score differences between PRIME$_{piecewise}$ and PRIME$_{affine}$ on full length set. The horizontal axis denotes reference alignment ID, and the vertical axis, the difference in sum-of-pairs or column scores on respective alignments of the full length set using PRIME$_{piecewise}$ and PRIME$_{affine}$. A positive difference score of an alignment is an indication that PRIME$_{piecewise}$ shows better performance than PRIME$_{affine}$ for the alignment, and vice versa.

Table 3: Average sum-of-pairs scores of full length set

	Ref. 1.1	Ref. 1.2	Ref. 2	Ref. 3	Ref. 4	Ref. 5	Overall	Ranksum
PRIME$_{piecewise}$	0.643	0.933	0.922	0.859	0.910	0.882	0.861	809
PRIME$_{affine}$	0.635	0.931	0.898	0.851	0.882	0.871	0.846	912
Prrn	0.574	0.923	0.901	0.820	0.859	0.821	0.821	1055
MAFFT	0.671	0.938	0.923	0.852	0.918	0.892	0.868	656
ProbCons	0.648	0.942	0.905	0.835	0.887	0.879	0.851	764
T-Coffee	0.613	0.933	0.916	0.826	0.900	0.858	0.846	884
MUSCLE	0.570	0.909	0.888	0.808	0.857	0.839	0.815	1260
DIALIGN-T	0.489	0.888	0.859	0.744	0.817	0.780	0.768	1668
POA	0.474	0.857	0.857	0.733	0.805	0.754	0.753	1804
ClustalW	0.497	0.864	0.848	0.722	0.786	0.713	0.748	1682

Each column shows average sum-of-pairs scores using all alignments of each reference of the full length set. Overall and Ranksum columns show the average sum-of-pairs scores and the rank sum of the Friedman test using all alignment of the whole full length set, respectively. A smaller rank sum means better accuracy.

Table 4: Average column scores of full length set

	Ref. 1.1	Ref. 1.2	Ref. 2	Ref. 3	Ref. 4	Ref. 5	Overall	Ranksum
PEIME$_{piecewise}$	0.416	0.839	0.445	0.566	0.573	0.552	0.572	846
PRIME$_{affine}$	0.391	0.826	0.413	0.539	0.483	0.496	0.531	958
Prrn	0.334	0.791	0.406	0.469	0.491	0.411	0.499	1080
MAFFT	0.449	0.839	0.436	0.560	0.607	0.544	0.583	759
ProbCons	0.401	0.851	0.374	0.462	0.530	0.509	0.532	847
T-Coffee	0.324	0.832	0.384	0.459	0.563	0.534	0.525	1017
MUSCLE	0.313	0.795	0.343	0.380	0.460	0.408	0.465	1246
DIALIGN-T	0.246	0.723	0.290	0.347	0.462	0.389	0.423	1554
POA	0.224	0.678	0.265	0.343	0.413	0.323	0.389	1690
ClustalW	0.221	0.707	0.219	0.271	0.404	0.237	0.368	1497

Each column shows average column scores using all alignments of each reference of the full length set. Overall and Ranksum columns show the average column scores and the rank sum of the Friedman test using all alignment of the whole full length set, respectively. A smaller rank sum means better accuracy.

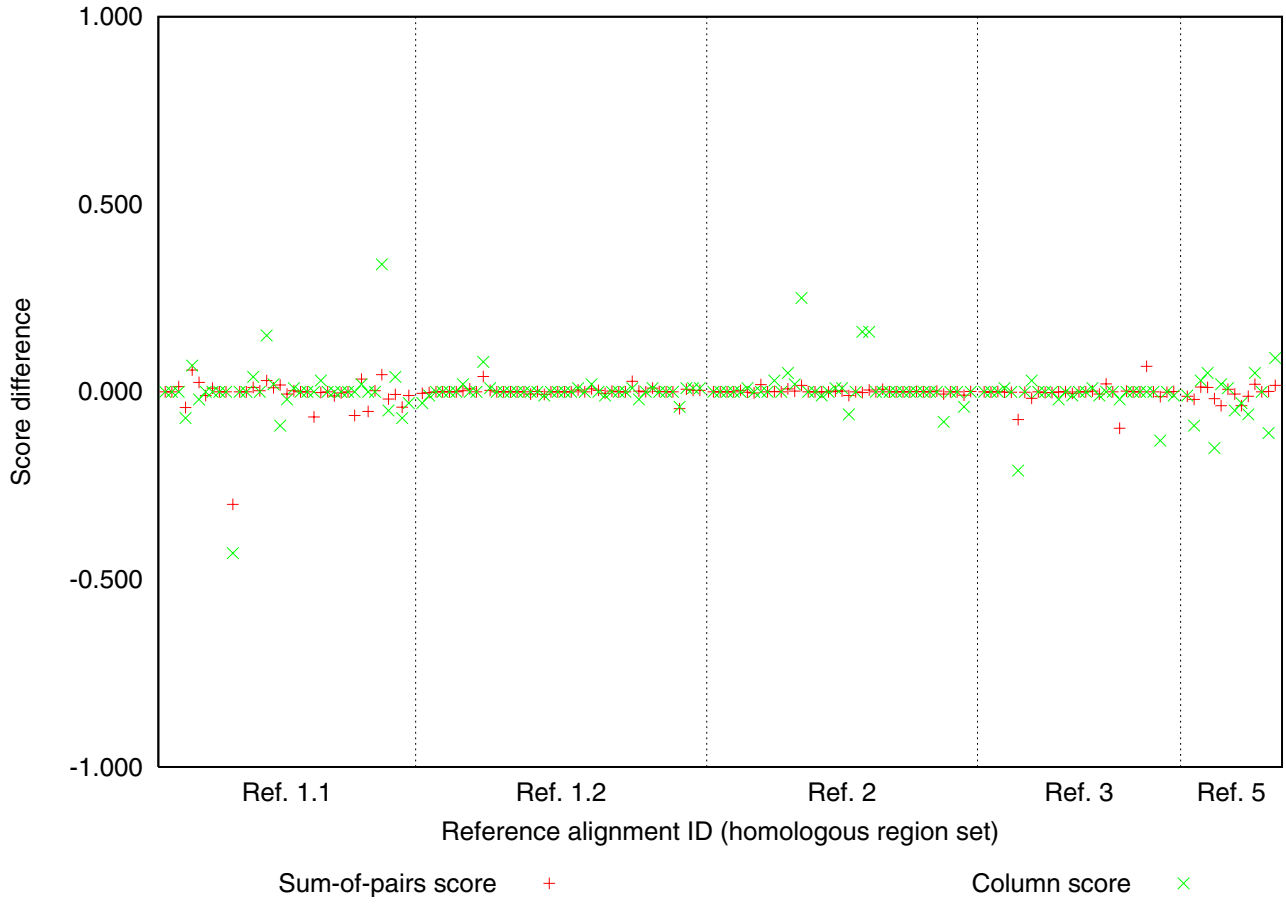

Figure 3
Score differences between PRIME$_{piecewise}$ and PRIME$_{affine}$ on homologous region set. The horizontal axis denotes reference alignment ID, and the vertical axis, the difference in sum-of-pairs or column scores on respective alignments of the homologous region set using PRIME$_{piecewise}$ and PRIME$_{affine}$. A positive difference score of an alignment is an indication that PRIME$_{piecewise}$ shows better performance than PRIME$_{affine}$ for the alignment, and vice versa.

Table 5: Average sum-of-pairs scores of homologous region set

	Ref. 1.1	Ref. 1.2	Ref. 2	Ref. 3	Ref. 5	Overall	Ranksum
$PRIME_{piecewise}$	0.772	0.940	0.955	0.903	0.891	0.894	613
$PRIME_{affine}$	0.781	0.938	0.954	0.907	0.896	0.897	634
Prrn	0.763	0.936	0.954	0.894	0.887	0.889	698
MAFFT	0.753	0.940	0.946	0.890	0.897	0.886	654
ProbCons	0.788	0.953	0.953	0.910	0.907	0.904	489
T-Coffee	0.704	0.939	0.940	0.878	0.888	0.870	821
MUSCLE	0.735	0.931	0.943	0.882	0.870	0.875	907
DIALIGN-T	0.573	0.901	0.897	0.793	0.821	0.798	1406
POA	0.634	0.877	0.923	0.822	0.800	0.816	1370
ClustalW	0.664	0.905	0.922	0.816	0.788	0.827	1263

Each column shows average sum-of-pairs scores using all alignments of each reference of the homologous region set. Overall and Ranksum columns show the average sum-of-pairs scores and the rank sum of the Friedman test using all alignment of the whole homologous region set, respectively. A smaller rank sum means better accuracy.

somewhat slower than most programs tested. The computational speed would be significantly improved by incorporating anchoring heuristics and refining source codes.

Discussions and Conclusion

The group-to-group sequence alignment algorithm is the key to most heuristic MSA algorithms. Although many group-to-group sequence alignment algorithms focus on position-specific gap opening penalties [7,8,10,12], they use essentially a constant gap extension penalty similar to that of the affine gap cost. For global MSA algorithms, use of the constant gap extension penalty could lead to deterioration of alignment accuracy when some of the sequences to be aligned have long indels. To our knowledge, POA version 2 [5] is the sole precedent that incorporates length-dependent gap extension penalties into the group-to-group sequence alignment algorithm. Examination of POA with various options indicated that length-dependent gap extension penalties with global alignment strategy are effective to improve alignment accuracy when some of the sequences to be aligned have long indels (data not shown).

In this paper, we proposed a novel group-to-group sequence alignment algorithm with the piecewise linear gap cost, and developed a program called PRIME. The advantage of using the piecewise linear gap cost is that this gap cost more accurately models the occurrence of long gap in a simple way than other gap cost does. As a result of BAliBASE benchmark test, PRIME achieved alignment accuracies comparable to the most accurate programs available today including L-INS-i of MAFFT, ProbCons, and T-Coffee. Unlike others, PRIME does not rely on pairwise alignment information. This implies that the introduction of length-dependent gap extension penalties could contribute to improving the alignment accuracy even when pairwise alignment information is not used.

It should be noted that our proposed algorithm has two inherent drawbacks. First, it is considerably slower than many popular algorithms. Second, selecting the parameters of the piecewise linear gap cost is somewhat more complicated than that of the affine gap cost. However, these drawbacks would not be serious enough to reduce the advantages of our proposed algorithm and PRIME.

Table 6: Average column scores of homologous region set

	Ref. 1.1	Ref. 1.2	Ref. 2	Ref. 3	Ref. 5	Overall	Ranksum
$PEIME_{piecewise}$	0.588	0.849	0.595	0.636	0.575	0.665	640
$PRIME_{affine}$	0.589	0.847	0.582	0.648	0.593	0.666	640
Prrn	0.561	0.834	0.601	0.630	0.558	0.654	708
MAFFT	0.552	0.846	0.532	0.631	0.578	0.640	670
ProbCons	0.591	0.875	0.540	0.625	0.583	0.658	548
T-Coffee	0.476	0.840	0.491	0.625	0.540	0.607	822
MUSCLE	0.496	0.823	0.496	0.574	0.501	0.596	946
DIALIGN-T	0.338	0.761	0.370	0.452	0.429	0.485	1339
POA	0.390	0.712	0.424	0.459	0.371	0.493	1348
ClustalW	0.416	0.791	0.443	0.475	0.394	0.529	1193

Each column shows average column scores using all alignments of each reference of the homologous region set. Overall and Ranksum columns show the average column scores and the rank sum of the Friedman test using all alignment of the whole homologous region set, respectively. A smaller rank sum means better accuracy.

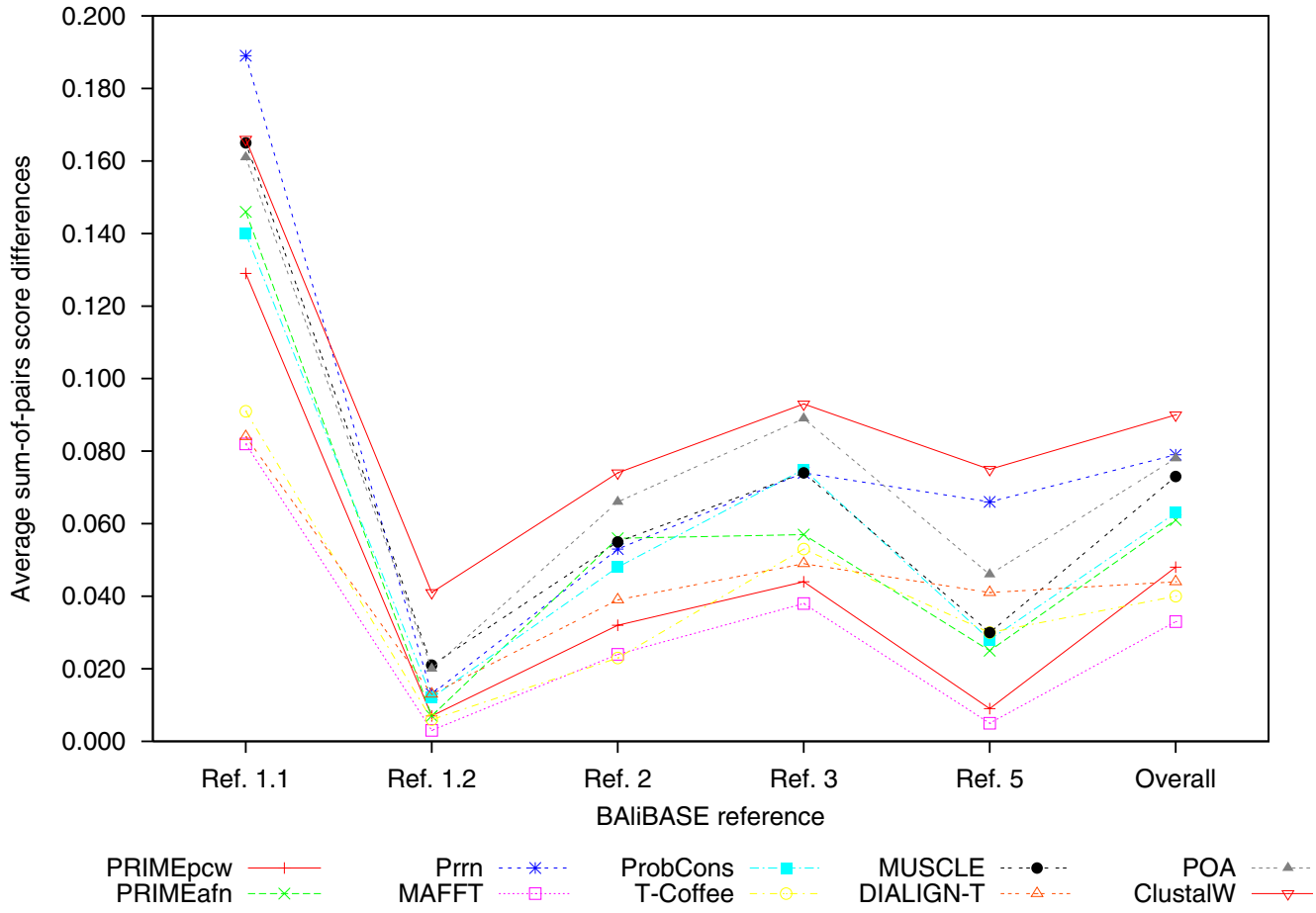

Figure 4

Average sum-of-pairs score differences between full length and homologous region sets. Each point means average sum-of-pairs score difference in respective alignments on each reference of the full length and homologous region sets. PRIME$_{pcw}$ denotes PRIME$_{piecewise}$, and PRIME$_{afn}$, PRIME$_{affine}$. The smaller absolute value of a score indicates that the introduction of long terminal indels less affects the alignment accuracy of a program.

Unlike most popular algorithms, PRIME can circumvent the time-consuming process for obtaining pairwise alignment information, and hence it is theoretically advantageous for aligning a large number of sequences. Our preliminary examination confirmed the expected dependency of computational time on the number of sequences to be aligned. However, the current version of PRIME is still slower than most of other programs over the examined range of the number of sequences (data not shown). One of the reasons is that the current PRIME does not use any heuristics, such as anchoring or grouping method used in Prrn, for reducing the computation. To improve calculation speed of PRIME without a loss of accuracy, we are attempting to incorporate these heuristics. To further improve alignment accuracy, we will investigate several problems including the conditions under which PRIME$_{affine}$ constructs more accurate alignment than PRIME$_{piecewise}$, the potential of other objective functions, and the effect of incorporating pairwise alignment information.

Availability and Requirements

Project name: PRIME project

Project home page: http://prime.cbrc.jp/

Operating system(s): Platform independent

Programming language: C++

Licence: GNU GPL version 2 or later

Any restrictions to use by non-academics: None

Table 7: Average quality scores of PREFAB

	Main		Weighting		Long gap	
	QS	Ranksum	QS	Ranksum	QS	Ranksum
PRIME$_{piecewise}$	0.721	8151	0.649	588	0.658	1408
PRIME$_{affine}$	0.718	8355	0.637	617	0.651	1504
Prrn	0.722	8120	0.624	621	0.653	1455
MAFFT	0.722	7744	0.639	585	0.660	1352
ProbCons	0.705	8659	0.620	594	0.637	1443
T-Coffee	0.700	9126	0.627	584	0.631	1640
MUSCLE	0.680	10446	0.607	642	0.596	1918
DIALIGN-T	0.621	13277	0.587	754	0.541	2506
POA	0.603	14662	0.554	868	0.513	2789
ClustalW	0.617	12952	0.603	650	0.519	2583
PSA$_{piecewise}$	0.591	14525	0.638	627	0.498	2804
PSA$_{affine}$	0.581	14789	0.621	670	0.489	2856

Each QS and Ranksum columns show the average quality scores and the rank sum of the Friedman test using quality scores on all alignments of each reference set, respectively. A smaller rank sum means better accuracy.

Table 8: Computation time

	BAliBASE	PREFAB
PRIME$_{piecewise}$	9.4×10^5	5.5×10^5
PRIME$_{affine}$	4.9×10^5	4.3×10^5
Prrn	6.8×10^5	1.9×10^5
MAFFT	1.9×10^4	2.7×10^4
ProbCons	6.4×10^4	1.9×10^5
T-Coffee	7.2×10^5	2.0×10^6
MUSCLE	7.9×10^3	1.6×10^4
DIALIGN-T	3.0×10^4	1.2×10^5
POA	1.0×10^4	2.6×10^4
ClustalW	8.3×10^3	2.7×10^4

BAliBASE column shows total times (sec.) of constructing all alignments of the full length and homologous region sets by each program, while PREFAB column, those of calculating whole alignments of the main and weighting sets only.

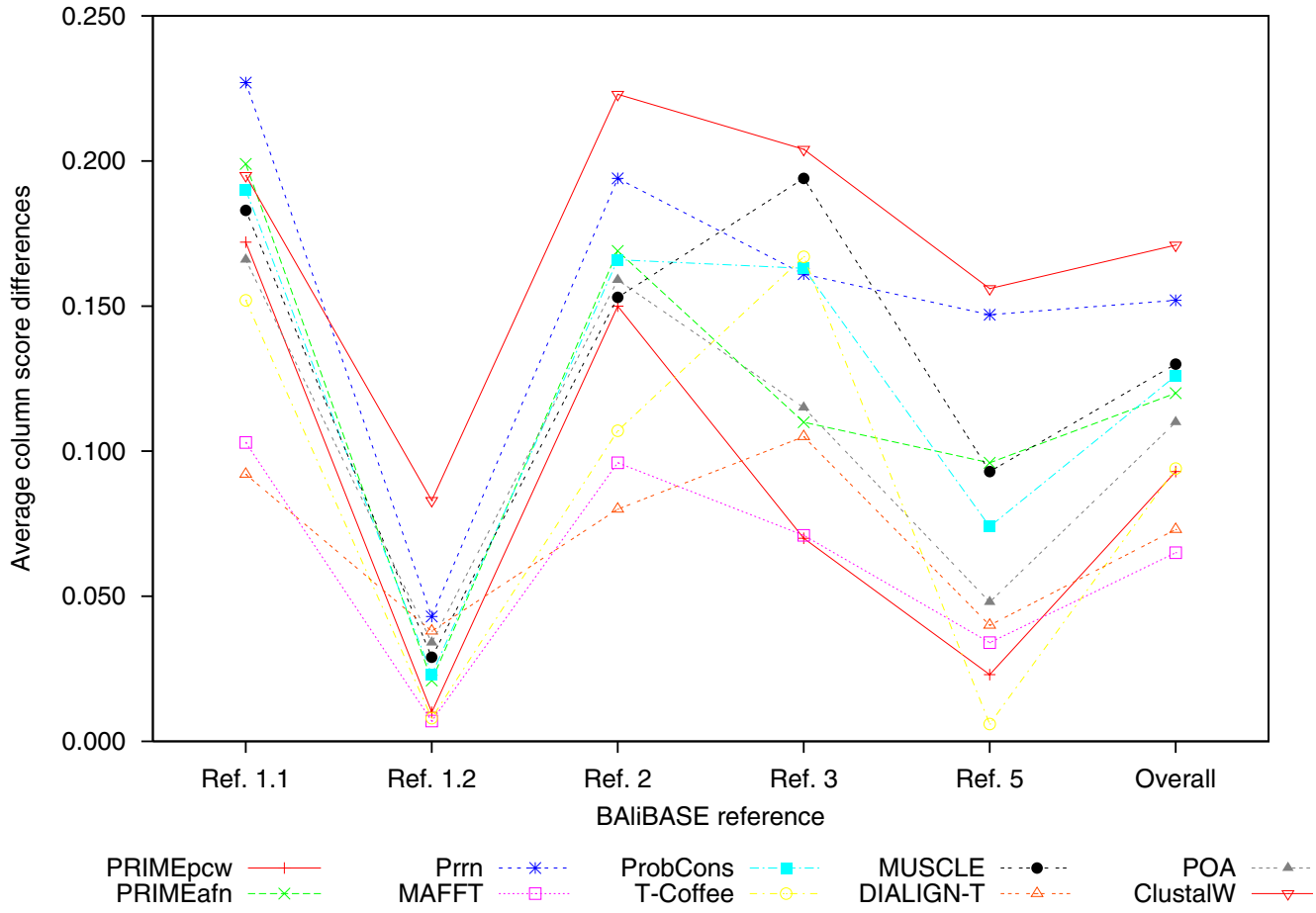

Figure 5

Average column score differences between full length and homologous region sets. Each point means average column score difference in respective alignments on each reference of the full length and homologous region sets. PRIME$_{pcw}$ denotes PRIME$_{piecewise}$, and PRIME$_{afn}$, PRIME$_{affine}$. The smaller absolute value of a score indicates that the introduction of long terminal indels less affects the alignment accuracy of a program.

Authors' contributions

SY devised the proposed algorithms, implemented PRIME, carried out the benchmark test and its evaluation, and drafted the manuscript. OG conceived of the study, devised the proposed algorithms, helped to implement PRIME, and drafted the manuscript. HY participated in the design and coordination of the study, and helped to implement PRIME. All authors read and approved the final manuscript.

Additional material

Additional File 1

***PRIME source code**. This tar.gz archive includes the source files of PRIME. To compile PRIME, one can check 'INSTALL' and 'Makefile' in the archive. Although 'Makefile' basically assumes GNU make and g++, another compiler can be used with a few modification of 'Makefile'.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-7-524-S1.gz]

Additional File 2

p-*values of the Friedman test of full length set*. *Each value is* p-*value of the Friedman test, indicating the significance of a difference in performance between programs of a row and a column. The upper right and lower left* p-*values are calculated using sum-of-pairs and column scores on all alignments of the whole full length set, respectively. $PRIME_{pcw}$ denotes $PRIME_{piecewise}$, and $PRIME_{afn}$, $PRIME_{affine}$. The respective signs + and - denote that a program of a row performs significantly better and worse than that of a column.*

Click here for file

[http://www.biomedcentral.com/content/supplementary/1471-2105-7-524-S2.pdf]

Additional File 3

p-*values of the Friedman test of homologous region set*. *Each value is* p-*value of the Friedman test, indicating the significance of a difference in performance between programs of a row and a column. The upper right and lower left* p-*values are calculated using sum-of-pairs and column scores on all alignments of the whole homologous region set, respectively. $PRIME_{pcw}$ denotes $PRIME_{piecewise}$, and $PRIME_{afn}$, $PRIME_{affine}$. The respective signs + and - denote that a program of a row performs significantly better and worse than that of a column.*

Click here for file

[http://www.biomedcentral.com/content/supplementary/1471-2105-7-524-S3.pdf]

Acknowledgements

This work was partially supported by a Grant-in-Aid for Scientific Research on Priority Areas (C) "Genome Information Science" from the Ministry of Education, Culture, Sports, Science and Technology of Japan; and by the Institute for Bioinformatics Research and Development (BIRD) of Japan Science and Technology Agency (JST).

References

1. Gotoh O: **Multiple sequence alignment: algorithms and applications.** *Adv Biophys* 1999, **36:**159-206.
2. Notredame C: **Recent progress in multiple sequence alignment: a survey.** *Pharmacogenomics* 2002, **3:**131-144.
3. Gotoh O, Yamada S, Yada T: **Multiple sequence alignment.** In *Handbook of computational molecular biology, Computer and Information Science Series* Edited by: Aluru S. Chapman & Hall/CRC; 2005:3–1-3–36.
4. Jiang T, Wang L: **Algorithmic methods for multiple sequence alignment.** In *Current topics in computational molecular biology* Edited by: Jiang T, Xu Y, Zhang MQ. Tsinghua University Press; 2002.
5. Grasso C, Lee C: **Combining partial order alignment and progressive multiple sequence alignment increases alignment speed and scalability to very large alignment problems.** *Bioinformatics* 2004, **20:**1546-1556.
6. Notredame C, Higgins DG, Heringa J: **T-Coffee: A novel method for fast and accurate multiple sequence alignment.** *J Mol Biol* 2000, **302:**205-217.
7. Thompson JD, Higgins DG, Gibson TJ: **CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice.** *Nucleic Acids Res* 1994, **22:**4673-4680.
8. Edgar RC: **MUSCLE: multiple sequence alignment with high accuracy and high throughput.** *Nucleic Acids Res* 2004, **32:**1792-1797.
9. Gotoh O: **Significant improvement in accuracy of multiple protein sequence alignments by iterative refinement as assessed by reference to structural alignments.** *J Mol Biol* 1996, **264:**823-838.
10. Katoh K, Misawa K, Kuma K, Miyata T: **MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform.** *Nucleic Acids Res* 2002, **30:**3059-3066.
11. Kececioglu J, Starrett D: **Aligning alignments exactly.** In *Proceedings of the Eighth Annual International Conference on Research in Computational Molecular Biology: 27–31 March 2004* Edited by: Bourne PE. San Diego, ACM; 2004:85-96.
12. Gotoh O: **Optimal alignment between groups of sequences and its application to multiple sequence alignment.** *Comput Appl Biosci* 1993, **9:**361-370.
13. McClure MA, Vasi TK, Fitch WM: **Comparative analysis of multiple protein-sequence alignment methods.** *Mol Biol Evol* 1994, **11:**571-592.
14. Thompson JD, Plewniak F, Poch O: **A comprehensive comparison of multiple sequence alignment programs.** *Nucleic Acids Res* 1999, **27:**2682-2690.
15. Katoh K, Kuma K, Toh H, Miyata T: **MAFFT version 5: improvement in accuracy of multiple sequence alignment.** *Nucleic Acids Res* 2005, **33:**511-518.
16. Do CB, Mahabhashyam MSP, Brudno M, Batzoglou S: **ProbCons: Probabilistic consistency-based multiple sequence alignment.** *Genome Res* 2005, **15:**330-340.
17. Miller W, Myers EW: **Sequence comparison with concave weighting functions.** *Bull Math Biol* 1988, **50:**97-120.
18. Gotoh O: **Optimal sequence alignment allowing for long gaps.** *Bull Math Biol* 1990, **52:**359-373.
19. Gotoh O: **Further improvement in methods of group-to-group sequence alignment with generalized profile operations.** *Comput Appl Biosci* 1994, **10:**379-387.
20. Gotoh O: **A weighting system and algorithm for aligning many phylogenetically related sequences.** *Comput Appl Biosci* 1995, **11:**543-551.
21. Gotoh O: **An improved algorithm for matching biological sequences.** *J Mol Biol* 1982, **162:**705-708.
22. Bahr A, Thompson JD, Thierry JC, Poch O: **BAliBASE (Benchmark Alignment dataBASE): enhancements for repeats, transmembrane sequences and circular permutations.** *Nucleic Acids Res* 2001, **29:**323-326.
23. Thompson JD, Koehl P, Ripp R, Poch O: **BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark.** *Proteins* 2005, **61:**127-136.
24. **PRIME** [http://prime.cbrc.jp/]
25. Thompson JD, Plewniak F, Poch O: **BAliBASE: a benchmark alignment database for the evaluation of multiple alignment programs.** *Bioinformatics* 1999, **15:**87-88.
26. Karplus K, Hu B: **Evaluation of protein multiple alignments by SAM-T99 using the BAliBASE multiple alignment test set.** *Bioinformatics* 2001, **17:**713-720.
27. O'Sullivan O, Suhre K, Abergel C, Higgins DG, Notredame C: **3DCoffee: combining protein sequences and structures within multiple sequence alignments.** *J Mol Biol* 2004, **340:**385-395.
28. Subramanian AR, Weyer-Menkhoff J, Kaufmann M, Morgenstern B: **DIALIGN-T: an improved algorithm for segment-based multiple sequence alignment.** *BMC Bioinformatics* 2005, **6:**66.