An Efficient Algorithm for Multiple Sequence Alignment *

Kuen-Feng Huang, Chang-Biau Yang and Kuo-Tsung Tseng
Department of Computer Science and Engineering
National Sun Yat-Sen University
cbyang@cse.nsysu.edu.tw

Abstract

The multiple sequence alignment (MSA) is a fundamental technique of molecular biology. Biological sequences are aligned with each other vertically in order to show the similarities and differences among them. In this paper, we first propose an efficient group alignment method to perform the alignment between two groups of sequences. Its time complexity is \( O(mnL_1L_2) \), where \( m \) and \( n \) are the number of sequences in the two groups, \( L_1 \) and \( L_2 \) are the length of the sequences in the two groups. Then we propose a clustering method to build the tree topology for merging, which is a top-down heuristics. The clustering method is based on the concept that the two sequences having the longest distance should be split into two clusters. The time complexity of our MSA algorithm is \( O(n^2L^2) \), where \( n \) is the number of sequences and \( L \) is the maximum length of all sequences. By our experiments, both the alignment quality and required time of our algorithm are better than Clustal W algorithm (using the neighboring joining method).

1 Introduction

Multiple sequence alignment (MSA) is important in functional, structural and evolutionary studies of biological sequences [3, 6, 8]. Due to its importance, many algorithms have been proposed [9, 11, 19, 21, 22]. The multiple sequence alignment problem is to obtain the alignment of a set of sequences with the best score based on some given scoring criteria [6]. A biological sequence is a string consisting of characters chosen from a set of alphabets \( \Sigma \), where \( \Sigma \) contains the 4 nucleotides \{A, C, G, T\} for nucleic acid sequences, or \( \Sigma \) contains the 20 amino acids \{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V\} for protein sequences. The gap is represented by the dash symbol (-).

The simplest case of multiple sequence alignment involves only two sequences. The basic idea of the multiple sequence alignment is to construct a global alignment so as to maximize similarities, or minimize distance, among all sequences. The dynamic programming is well-known for the alignment problems. Finding the optimal alignment with dynamic programming for a pair of sequences can be done in \( O(n^2) \) time [20], where \( n \) is the length of the two strings. Unfortunately, for the general problem of aligning \( k \) sequences, \( O(n^k) \) time is required [13].

Many criteria were proposed to measure the goodness of various multiple sequence alignments. One of the popular criteria is the sum of pairs. It generates a score for each pairwise alignment with a Dayhoff [7] or Blosum [2] matrix, and creates a score for the multiple sequence alignment by summing all scores of the pairwise alignments.

In this paper, we propose a clustering method and a group alignment method for multiple sequence alignment as a practical method. In the group alignment method, given two disjoint sets (groups) of sequences, we want to merge them into one group, so that each gap is inserted simultaneously in the same position of each sequence in one group. The time complexity of our group alignment is \( O(mnL_1L_2) \), where \( m \) and \( n \) are the number of sequences in the two groups, \( L_1 \) and \( L_2 \) are the length of the sequences in the two groups. As the clustering method, the main idea is that two sequences with the longest distance should be put in two clusters, and its time complexity is \( O(n^3L^2) \), where \( n \) is the number of sequences and \( L \) is the maximum length of all sequences. The experimental results show that both the alignment quality and required time of our algorithms are better than Clustal W algorithm (using the neighboring joining method).

*This research work was partially supported by the National Science Council of the Republic of China under contract NSC-89-2418-E-110-021.
The rest of this paper is organized as follows. Section 2 gives the definition of the multiple sequence alignment problem, a brief survey on some previous algorithms and the complexity of the MSA problem. Our group alignment method is given in Section 3. We present our new approach based on the clustering method in Section 4 and our experimental results are shown in Section 5. Finally Section 6 ends the paper with our conclusion and future works.

2 Preliminary

There are two kinds of multiple sequence alignment. One is the global alignment [15], which constructs an alignment throughout the whole sequences. It seeks to line up the sequences so that the similarity in the alignment is maximized. The other is the local alignment [2], which only attempts to identify an ordered series of motifs, or homologous regions. It seeks to line up a subsequence from the sequences such that they yield the highest scoring region. In this paper, we study only the global alignment.

A (global) multiple sequence alignment of a set of sequences is obtained by inserting gap character ‘-’ into each sequence so that all resulting sequences have the same length and no column has only gap character.

Definition 1 Given strings $S_1, S_2, \ldots, S_k$, a (global) multiple alignment maps them to strings $A_1, A_2, \ldots, A_k$ that may contain spaces, where

1. $|A_1| = |A_2| = \ldots = |A_k|$,
2. $A_i$ by removing all ”-“ gap characters is equal to $S_i$,
3. no column has only gap character.

We want to find the alignment with the best score, where the score of MSA is the sum of the scores of all columns. In a pairwise alignment, we simply sum the similarity scores of corresponding characters. The score of the MSA is the sum of the scores of all columns. There are two kinds of input sequence data: DNA sequence and protein sequence. The simplest cost function for two DNA sequences is the edit distance as follows:

$$\delta(x, y) = \begin{cases} c_1 & \text{if } x = y, \\ c_2 & \text{if } x \neq y, \\ c_3 & \text{if } x = \text{gap or } y = \text{gap}, \end{cases}$$

where $c_1$ and $c_2$ denote the cost of one match and one substitution, respectively, and $c_3$ denotes the value of affine gap cost [1].

For the protein sequences, a substitution matrix $DM$ are used as follows:

$$\delta(x, y) = \begin{cases} DM(x, y) & \text{if } x \neq \text{gap and } y \neq \text{gap}, \\ 0 & \text{if } x = \text{gap and } y = \text{gap}, \\ \text{affine gap cost} & \text{if } x = \text{gap or } y = \text{gap}. \end{cases}$$


In the multiple sequence alignment, there are various scoring methods aimed at satisfying various goals. Formally, we can express the multiple sequence alignment problem as an optimization problem. Sum-of-pairs (SP) measure, tree alignment and star alignment are three metrics used to evaluate each column. The SP, tree alignment and star alignment costs for the same column may be very different [3].

In the star alignment, the score is the sum of the pairwise alignments between all sequences and the chosen consensus sequence (center sequence). In the tree alignment, it is also possible to compute score in an evolutionary tree by summing up the pairwise score on each edge in the tree, which represents the evolutionary distance between two nodes. Hence, this scoring method tries to explain the evolution process by minimizing the evolutionary distance.

The formal definition of the SP measure is as follows:

Definition 2 The sum-of-pairs score for a multiple sequence alignment $A$ of $k$ sequences is the sum of the scores of all $\binom{k}{2}$ pairwise alignments of $A$.

Figure 1 shows an example of the above scoring rule: If a column has two identical characters, it receives value 0; otherwise, it receives value 1.

A gap is caused by a mutation which removes a sequence of residues. Indels (insertions or deletions) should appear less frequently. Therefore a long gap is often more suitable than several segments of gaps.

The penalty for a gap has two parts: a penalty $P_g$ is related to the initiation of a gap, and another penalty $P_c$ is related to the length of a gap. That is, the gap penalty is $P_g + kP_c$, called affine gap penalty, where $P_g$ and $P_c$ are both constants, $P_g \geq 0$, $P_c \geq 0$, and $k \geq 1$ is the length of the gap.

The problem of finding an alignment with an affine gap penalty can also be solved by the dynamic programming approach [6]. Suppose the
two input sequences are $a_1a_2\ldots a_n$ and $b_1b_2\ldots b_m$. Some notations are given as follows.

1. $A(i, j)$ is the score of an optimal alignment of $a_1a_2\ldots a_i$ and $b_1b_2\ldots b_j$.

2. $V(i, j)$ is the score of an optimal alignment of $a_1a_2\ldots a_i$ and $b_1b_2\ldots b_j$ whose last pair matches $a_i$ with $b_j$.

3. $D(i, j)$ is the score of an optimal alignment of $a_1a_2\ldots a_i$ and $b_1b_2\ldots b_j$ whose last pair matches a space with $b_j$.

4. $I(i, j)$ is the score of an optimal alignment of $a_1a_2\ldots a_i$ and $b_1b_2\ldots b_j$ whose last pair matches a space with $b_j$.

The dynamic programming method has the following initialization:

$$A(0, 0) = 0,$$
$$A(i, 0) = -P_g - iP_e, \text{ for } i > 0,$$
$$A(0, j) = -P_g - jP_e, \text{ for } j > 0,$$
$$D(i, 0) = -\infty, \text{ for } i > 0,$$
$$I(0, j) = -\infty, \text{ for } j > 0.$$ 

Then the following computations for $i > 0$ and $j > 0$ are performed:

$$A(i, j) = \max \{V(i, j), D(i, j), I(i, j)\},$$
$$V(i, j) = A(i - 1, j - 1) + \delta(a_i, b_j),$$
$$D(i, j) = \max \{D(i - 1, j) - P_e, A(i - 1, j) - P_g - P_e\},$$
$$I(i, j) = \max \{I(i, j - 1) - P_e, A(i, j - 1) - P_g - P_e\}.$$

The SP alignment and the tree alignment on a fixed binary tree topology have been shown to be NP-complete [13]. A polynomial time approximation scheme has been presented [12]. An optimal alignment of $k$ sequences can be obtained by dynamic programming in $O((2n)^k)$ with $O(n^k)$ space [13], where $n$ is the maximum length over all sequences. The algorithms proposed by Carillo and Lipman [5] and Kececioglu [14] can reduce the time for finding an exact solution by reducing the search space with an upper bound.

For the SP alignment, Gusfield [10] presented a $(2 - 2/k)$-approximation algorithm, and Pevsner [16] reduced the approximation factor to $(2 - 3/k)$.

Bafna et al. [4] presented a $(2 - l/k)$-approximation algorithm, for any fixed constant $l$. For the tree alignment, Gusfield [10] showed that the cost of the minimum spanning tree of the sequences is at most twice the cost of optimal multiple alignment.

For a fixed tree topology, Jiang and Lawler et al. [12] presented a $(1 + \epsilon)$-approximation scheme with polynomial time for small $\epsilon$.

3 The Group Alignment Method

In this section, we propose our group alignment method, and give an example to show how our algorithm works. Given two sets of sequences, in which each set of sequences have been aligned (by some other algorithms or ours), our group alignment method is to combine the two sets into one set and align all sequences. Our main idea is that each gap is inserted simultaneously in the same position of each sequence in one group. The method is as follows.

**Algorithm: Group Alignment**

**Input:** Two alignments $X = \{X_1, X_2, \ldots, X_m\}$ and $Y = \{Y_1, Y_2, \ldots, Y_n\}$, where each $X_k$, $1 \leq k \leq m$, or $Y_l$, $1 \leq l \leq n$, denotes one sequence with some possible gaps within it, created by the alignment in its group(s).

**Output:** A multiple alignment of $X$ and $Y$.
Notations: Let sequence $X_k$, $1 \leq k \leq m$, be denoted as $X_k, X_{k_2} \cdots X_{k_n}$, and sequence $Y_l$, $1 \leq l \leq n$, be denoted as $Y_l, Y_{l_2} \cdots Y_{l_m}$, where $L_1$ and $L_2$ are the lengths of each sequence in $X$ and $Y$, respectively. $w()$ denotes an entry of the score matrix (such as PAM-250), we are using and $\alpha$ denotes affine gap penalty, which can be adjusted.

**Step 1:** Compute the following:
\[
D_{i,j} = \min \left\{ \sum_{l=1}^{m} w(X_{k_l}, -), \sum_{l=1}^{m} w(X_{k_l}, -) + \alpha \right\},
\]
\[
I_{i,j} = \min \left\{ \sum_{l=1}^{n} w(-, Y_{l_j}), \sum_{l=1}^{n} w(-, Y_{l_j}) + \alpha \right\},
\]
\[
T_{i,j} = \min \left\{ \sum_{l=1}^{m} \sum_{l=1}^{n} w(X_{k_l}, Y_{l_j}), \sum_{l=1}^{m} \sum_{l=1}^{n} w(X_{k_l}, Y_{l_j}) + \alpha \right\},
\]
where $1 \leq i \leq L_1$, $1 \leq j \leq L_2$.

**Step 2:** After $T_{i,j}$ has been found, we can trace back to find the multiple sequence alignment of $X$ and $Y$.

The time complexity of the above group alignment algorithm is $O(mnL_1L_2)$. The distance between a sequence $S_k$ and a set of sequences $G$, denoted as $d(S_k, G)$, is defined as the smallest among the distances between $S_k$ and all sequences in $G$.

We now consider the construction of the final multiple sequence alignment on Group $X$ and Group $Y$ ($X$, $Y$ have been aligned by our methods) by our group alignment method (Algorithm Group Alignment). Suppose that in the score function, $\alpha = 0$, the costs of a match, a mismatch and an indel are 0, 1 and 1, respectively.

**Group $X$**:
\[
(S_1) \text{ AAGGCTT}
\]
\[
(S_5) \text{ -AAGGCTT}
\]
\[
(S_3) \text{ -AGGGA-T}
\]

**Group $Y$**:
\[
(S_2) \text{ -CGATT}
\]
\[
(S_4) \text{ TCGA-}
\]

Clearly, $m=3$, $n=2$, $L_1 = 8$, $L_2 = 6$.

**Initial case:**
$T_{0,0} = 0$.
$T_{i,j} = T_{i-1,j} + \sum_{l=1}^{m} w(-, Y_{l_j}),$ \hspace{1cm} $i = 0, 1 \leq j \leq 6.$
$T_{i,j} = T_{i-1,j} + \sum_{l=1}^{m} w(X_{k_l}, -),$ \hspace{1cm} $1 \leq i \leq 8, j = 0.$

**Other case:**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>C</td>
<td>G</td>
<td>A</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>C</td>
<td>G</td>
<td>A</td>
<td>-</td>
</tr>
</tbody>
</table>

$\uparrow$: $3, 9, 15, 21, 24, 27$  
A: $2, 4, 9, 15, 19, 22, 25$  
A: $8, 8, 10, 15, 18, 21$  
G: $14, 14, 14, 10, 16, 19, 22$  
G: $20, 20, 20, 14, 16, 19, 22$  
C: $26, 26, 24, 24, 20, 22, 25$  
C: $32, 32, 28, 26, 24, 26, 28$  
T: $36, 35, 34, 30, 28, 27, 29$  
T: $42, 39, 40, 36, 34, 31, 30$

Figure 2: The group alignment method on Group $X(2)$ and Group $Y(3)$.

$T_{i,j} = \min \left\{ \sum_{l=1}^{m} \sum_{l=1}^{n} w(X_{k_l}, Y_{l_j}), \sum_{l=1}^{m} \sum_{l=1}^{n} w(X_{k_l}, -), \sum_{l=1}^{m} \sum_{l=1}^{n} w(-, Y_{l_j}) \right\}$.

The complete result of our group alignment is shown in Figure 2. The computations of some of the entries are shown as follows.

$T_{0,1} = T_{0,0} + 3w(-, -) + 3w(\alpha, -)$
$= 3.$

$T_{0,5} = T_{0,4} + 3w(-, \alpha) + 3w(-, -)$
$= 24.$

$T_{1,0} = T_{0,0} + 2w(\alpha, -) + 2w(-, -) + 2w(-, \alpha)$
$= 2.$

$T_{7,0} = T_{6,0} + 2w(\alpha, -) + 2w(\alpha, -) + 2w(-, \alpha)$
$= 36.$

$T_{0,0} + w(\alpha, -) + w(\alpha, \alpha) + 2w(-, \alpha) + 2w(-, \alpha)$

$T_{1,1} = \min \left\{ T_{0,1} + 2w(\alpha, -) + 4w(-, -), T_{1,0} + 3w(-, -) + 3w(-, \alpha), \right\}$
$= 4.$

$T_{0,1} + 2w(\alpha, \alpha) + 4w(-, \alpha)$

$T_{1,2} = \min \left\{ T_{0,2} + 2w(\alpha, -) + 4w(-, -), T_{1,1} + 3w(-, \alpha) + 3w(-, \alpha), \right\}$
$= 9.$
\[ T_{7,5} = \min \begin{cases} T_{6,4} + 2w(T, T) + 2w(T, -) + w(-, T) + w(-, -), \\ T_{6,5} + 2w(T, -) + 2w(T, -) + 2w(-, -), \\ T_{7,4} + 3w(-, T) + 3w(-, -), \end{cases} = 27. \]

In Figure 2, we can track back from the right lower corner to get the alignment. An up arrow means that a gap is inserted into all sequences of Group \( Y \) and a left arrow means that a gap is inserted into all sequences of Group \( X \). For example, when we track back from \( T_{8,6} \) to \( T_{7,5} \), it represents the following alignment:

\[
\text{Group } X \\
\text{T} \\
\text{T} \\
\text{T} \\
\text{Group } Y \\
\text{T} \\
\]

The alignment of backtracking from \( T_{6,4} \) to \( T_{5,3} \) is as follows:

\[
\text{Group } X \\
\text{CTT} \\
\text{CTT} \\
\text{A-T} \\
\text{Group } Y \\
\text{ATT} \\
\text{A--} \\
\]

Finally, backtracking from \( T_{4,0} \) to \( T_{0,0} \) indicates the following alignment:

\[
\text{Group } X \\
\text{AGGGCTT} \\
\text{--AGGGCTT} \\
\text{--AGGG-A-T} \\
\text{Group } Y \\
\text{--CG-ATT} \\
\text{--TCG-A--} \\
\]

The final multiple sequence alignment is given as follows:

\[
S_1 = \text{AGGGCTT} \\
S_2 = \text{--CG-ATT} \\
S_3 = \text{--AGGG-A-T} \\
S_4 = \text{--TCG-A--} \\
S_5 = \text{--AGGGCTT} \\
\]

4 The Clustering Method

Since our group alignment method is introduced, we would like to show our CMSA (Clustering Multiple Sequence Alignment) algorithm. The tree based method uses a technique of "once a gap, always a gap". Our main idea is to reduce the number of gaps in each group. Thus, if we put the two sequences of the longest distance in two distinct groups, we can get a better multiple sequence alignment when the input set of sequences are very similar. The detail of our CMSA is as follows.

**Algorithm: CMSA**

(Clustering Multiple Sequence Alignment)

**Input:** A set of sequences \( S = \{S_1, S_2, \ldots, S_n\} \).

**Output:** A multiple sequence alignment of \( S \).

**Step 1:** If \( |S| \leq 1 \), then stop.

**Step 2:** Compute the optimal alignment on each pair of sequences in \( S \). Then construct the distance matrix for \( S \).

**Step 3:** Sort all entries in the distance matrix into nonincreasing order.

**Step 4:** Create a set of sequences \( R = S \).

**Step 5:** In \( R \), select a pair of sequences \( S_i \) and \( S_j \) such that \( S_i \) and \( S_j \) have the longest distance.

**Step 6:** Let \( G_1 = \{S_i\} \) and \( G_2 = \{S_j\} \). \( R = R - \{S_i, S_j\} \).

Perform the following substeps until \( R \) becomes empty.

**Step 6.1:** Select \( S_k \in R \) such that \( \min\{d(S_k, G_1), d(S_k, G_2)\} \) is the minimum.

**Step 6.2:** If \( d(S_k, G_1) \leq d(S_k, G_2) \), then \( G_1 = G_1 \cup \{S_k\} \); otherwise \( G_2 = G_2 \cup \{S_k\} \).

**Step 6.3:** \( R = R - \{S_k\} \).

**Step 7:** Recursively apply this algorithm (Algorithm CMSA) by setting the input \( S = G_1 \).

Recursively apply this algorithm (Algorithm CMSA) by setting the input \( S = G_2 \).

**Step 8:** Perform our group alignment method (Algorithm Group Alignment) on \( G_1 \) and \( G_2 \).

Let \( L = \max\{|S_1|, |S_2|, \ldots, |S_n|\} \), where \( |S_i| \) denotes the length of \( S_i \). The complexity of each step is as follows.

**Step 2:** \( O(n^2L^2) \).

**Step 3:** \( O(n^2 \log n) \).

**Step 6:** \( O(n) \).

**Step 8:** \( O(n^2L^2) \).
Table 1: The distance matrix for five sequences.

<table>
<thead>
<tr>
<th></th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
<th>$S_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>-</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>$S_2$</td>
<td>-</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>-</td>
</tr>
<tr>
<td>$S_3$</td>
<td>-</td>
<td>-</td>
<td>4</td>
<td>2</td>
<td>-</td>
</tr>
<tr>
<td>$S_4$</td>
<td>-</td>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$S_5$</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The time required for one recursion is $O(n^2L^2)$, since $L > n$ in almost all practical cases. Combining with the recursive work in Step 7, we obtain the time complexity of the algorithm is $O(n^3L^2)$.

We now explain our algorithm step by step. Let us consider the following five sequences. Suppose that in the score function, $\alpha = 0$, the costs of a match, a mismatch and an indel are 0, 1 and 1, respectively.

$$S_1 = AAGGCTTG$$
$$S_2 = GATT$$
$$S_3 = AAGGAT$$
$$S_4 = TCGA$$
$$S_5 = AAGGCTTG$$

In Step 2, we obtain a distance matrix, as shown in Table 1. Our job is to divide the five sequences into two groups in Steps 5 and 6. The dividing process is shown in Figure 3. The first sequence put in each group is represented by a gray node. The number associated with each node represents the order that the sequence is added. The solid line denotes the distance between the sequence and that group. Figure 3 (b) and (c) are divided from Figure 3 (a). Initially, $G_1 = \{S_1\}$ and $G_2 = \{S_4\}$ because $S_1$ and $S_4$ have the longest distance. $S_5$ is the closest to $G_1$. Thus $S_5$ is added to $G_1$ and $G_1 = \{S_1, S_5\}$. We also find that $S_3$ is close (similar) to $S_5$. Thus $S_3$ is added to $G_1$ and $G_1 = \{S_1, S_3, S_5\}$. Finally, $S_2$ is close (similar) to $S_4$. Thus $S_2$ is added to $G_2$. The final clustering result is $G_1 = \{S_1, S_3, S_5\}$ and $G_2 = \{S_2, S_4\}$.

Since the number of sequences in $\{S_1, S_3, S_5\}$ is greater than 2, we divide $G_1$ again, as shown in Figure 3 (d) and (e). In Figure 3 (d) and (e), $G'_1 = \{S_1\}$ and $G'_2 = \{S_3\}$ because $S_1$ and $S_3$ have the longest distance. $S_5$ is the closest to $G'_1$. Thus $S_5$ is added to $G'_1$ and $G'_1 = \{S_1, S_5\}$. Now we have three groups $G'_1 = \{S_1, S_5\}$, $G'_2 = \{S_3\}$ and $G_2 = \{S_2, S_4\}$.

In Step 8, we perform our group alignment method to construct the multiple sequence alignment. The order of the alignment is as follows:

1. Align $S_1$ and $S_4$ in $G'_2$. The resulting alignment is denoted as Group 1.
2. Align $S_2$ with Group 1. The resulting alignment is denoted as Group 2($X$).
3. Align $S_3$ and $S_5$. The resulting alignment is denoted as Group 3($Y$).
4. Align Group 2($X$) and Group 3($Y$), as shown in Figure 2.

5 Experimental Results

In this section, we will show our experimental results and analyze the performance of our algorithm. Our algorithm is implemented by GCC on PC with AMD Duron processor 750 MHZ and 256 MB RAM. All test sequences are protein sequences and real biological sequences, that is, $|\sum| = 20$. We use the sum of pairs measure to determine the goodness of multiple sequence alignment. The score matrix we use is PAM-250, as shown in Table 2.

Clustal W is one of the famous software packages to do the multiple sequence alignment, and it can be found in the Internet [21]. Clustal W algorithm is a bottom-up method, while our clustering method is a top-down method. Clustal W applies the neighbor joining (NJ) algorithm [17] to construct a binary tree topology (clustering process), then uses the consensus alignment method with automatically turning the gap cost on the tree topology.
Table 2: PAM-250 score matrix.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
<th>P</th>
<th>Q</th>
<th>R</th>
<th>S</th>
<th>T</th>
<th>V</th>
<th>W</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>-2</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>-5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>-5</td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>-3</td>
<td>-4</td>
<td>-6</td>
<td>-5</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td>-5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>-1</td>
<td>-3</td>
<td>1</td>
<td>1</td>
<td>-2</td>
<td>-2</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>1</td>
<td>-3</td>
<td>-2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>J</td>
<td>-1</td>
<td>-5</td>
<td>0</td>
<td>0</td>
<td>-5</td>
<td>-2</td>
<td>0</td>
<td>-2</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>-2</td>
<td>-6</td>
<td>-4</td>
<td>-3</td>
<td>2</td>
<td>-4</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>-1</td>
<td>-5</td>
<td>-3</td>
<td>-2</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>-1</td>
<td>2</td>
<td>1</td>
<td>-3</td>
<td>0</td>
<td>2</td>
<td>-2</td>
<td>1</td>
<td>-3</td>
<td>-2</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>-3</td>
<td>-1</td>
<td>-1</td>
<td>-6</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>-3</td>
<td>-2</td>
<td>0</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Q</td>
<td>0</td>
<td>-5</td>
<td>2</td>
<td>2</td>
<td>-5</td>
<td>-1</td>
<td>3</td>
<td>-2</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>-2</td>
<td>-4</td>
<td>-1</td>
<td>-1</td>
<td>-4</td>
<td>-3</td>
<td>2</td>
<td>-2</td>
<td>3</td>
<td>-3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
<td>-3</td>
<td>-2</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>-2</td>
<td>0</td>
<td>0</td>
<td>-3</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>-2</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>-1</td>
<td>-2</td>
<td>4</td>
<td>-2</td>
<td>2</td>
<td>-2</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-1</td>
<td>0</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>-6</td>
<td>-8</td>
<td>-7</td>
<td>-7</td>
<td>0</td>
<td>-7</td>
<td>-3</td>
<td>-5</td>
<td>-3</td>
<td>-2</td>
<td>-4</td>
<td>-4</td>
<td>-6</td>
<td>-5</td>
<td>2</td>
<td>-2</td>
<td>-5</td>
<td>-6</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>Y</td>
<td>-3</td>
<td>0</td>
<td>-4</td>
<td>-4</td>
<td>7</td>
<td>-5</td>
<td>0</td>
<td>-1</td>
<td>-4</td>
<td>-1</td>
<td>-2</td>
<td>-2</td>
<td>-5</td>
<td>-4</td>
<td>-4</td>
<td>-3</td>
<td>-3</td>
<td>-2</td>
<td>0</td>
<td>10</td>
</tr>
</tbody>
</table>

In this paper, we propose one clustering method and one group alignment method. To obtain the performance of our algorithm, we compare three algorithms: (1) our algorithm, (2) Clustal W, and (3) the mixed algorithm. The mixed algorithm consists of the NJ algorithm and our group alignment. Thus, in our experimental results, we can see the performance of two aligning processes and two clustering processes.

Table 3 shows the scores and computing time of the three algorithms on real biological sequences. We can see that our algorithm is faster than Clustal W and the scores of our algorithm are better than those of Clustal W (We get worse scores in only few test cases). Comparing with the mixed algorithm, our algorithm spends almost the same time as the mixed algorithm and our algorithm has better scores. It can be concluded that our algorithm is more suitable for real biological sequences as compared with Clustal W algorithm. In addition, our group alignment method is more efficient than Clustal W. The possible reason is that we do not turn gap cost. The sources of real biological sequences are shown in Table 4.

6 Conclusion

In this paper, we proposed an efficient heuristics algorithm to solve the multiple sequence alignment problem, which consists of the clustering method and the group alignment method. Both of our algorithm and Clustal W algorithm are the tree based method. The main difference is that our algorithm is a top-down method, while Clustal W is a bottom-up method. In our algorithm, we first compare all sequences pairwise by the dynamic programming algorithm, then perform cluster analysis on the pairwise data to generate a tree hierarchy for alignment. This is a fast method, but it may produce low-quality alignment. The reason is that any error which occurs in the pairwise alignment cannot be corrected; it will be carried to the final alignment. Thus, our algorithm is more suitable for real biological sequences or very similar sequences.

Experimental results show that our method is better than the Clustal W for almost all sequences with high similarity. In some cases, it is still better than the Clustal W for the sequences with low similarity. To build a practical MSA software package, we should consider the way to turn gap penalties and choose proper score matrices, such as Dayhoff or Blosum matrix.

With dynamic programming, finding the optimal alignment for a pair of sequences can be done in \(O(L^2)\) time, where \(L\) is the length of the two strings. And, it is well known that for the general optimization problem of aligning \(n\) sequences of length \(L\), \(O(L^n)\) time is required. Based on
the idea of our group alignment method, we can
develop an algorithm to solve the optimal MSA
problem in $O(2^n L^2)$ time. We have written a pro-
gram to verify the algorithm and we do not find
any counter test case until now. We are now trying
to prove its correctness.

References

sequence alignment. *Journal of Theoretical Bi-

Myers, and D. J. Lipman. Basic local align-
ment search tool. *Journal of Molecular Bi-

stars and multiple sequence alignment. *SIAM
Journal on Applied Mathematics, 49(1):197–

proximation algorithms for multiple sequence
Combinatorial Matching*, volume 807, pages

sequence alignment problem in biology. *SIAM
Journal on Applied Mathematics, 48:1073–

Chiu. A survey of multiple sequence com-
parison methods. *Bulletin of Mathematical

and Structure*. National Biomedical Research


Table 4: The source of real biological test data.

<table>
<thead>
<tr>
<th>source of test data</th>
</tr>
</thead>
<tbody>
<tr>
<td>data1 Accession of NCBI: AF_065226, AF_065206, AF_065204, AF_065341</td>
</tr>
<tr>
<td>data2 Accession of NCBI: AF_065568, BA2023, AF_12135, AF_12006, AF_112106, AF_065565, AF_12122</td>
</tr>
<tr>
<td>data3 Accession of NCBI: AF_065226, AF_065206, AF_065204, AF_065431, AF_065228, AF_065666, AF_12122</td>
</tr>
<tr>
<td>data4 Accession of NCBI: AF_065435, AF_065215, AF_06228, BA05619, AKR0892, AKR0892</td>
</tr>
<tr>
<td>data5 Accession of NCBI: AK0R554, AK0R578, AF_071454, AF_078765, AF_07371, AF_08008, AF_07584</td>
</tr>
<tr>
<td>data6 Accession of NCBI: AK0R429, AK0R427, AKR052, AF_08164, AF_08292, AKR00049</td>
</tr>
<tr>
<td>data7 Accession of NCBI: AF_1224, AF_06565, AF_065222, AF_065228, AF_065228, AF_065228, AF_065228</td>
</tr>
<tr>
<td>data8 Accession of NCBI: AF_065222, AAG0660, AF_065221, AF_065428</td>
</tr>
<tr>
<td>data9 Accession of NCBI: AF_065588, AF_065474, AF_06528, AF_06569, AF_06542, AF_06225, AF_06212</td>
</tr>
<tr>
<td>data10 Accession of NCBI: AF_12227, AF_065692, AF_068342, AF_068225, AF_068212</td>
</tr>
<tr>
<td>data11 Accession of NCBI: AF_0624, AF_11431, AKR0892, ACF3838, ACF3838</td>
</tr>
<tr>
<td>data12 Accession of NCBI: AF_06522, AAG0660, AF_065221, AF_06520, AF_06542, AF_06396, AF_06825</td>
</tr>
<tr>
<td>data13 Accession of NCBI: AF_065396, AF_06825, AF_11491, AKR06638</td>
</tr>
<tr>
<td>data14 Accession of NCBI: AAG0854, AAG0574, AF_071042, AF_087892</td>
</tr>
<tr>
<td>data15 Accession of NCBI: AF_06555, AF_12132, BAB0348, AF_01378, BAB0335, BAB0206, BAB0207, BAB0207, BAB0207, BAB0207, BAB0207, BAB0207, BAB0207</td>
</tr>
<tr>
<td>data16 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data17 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data18 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data19 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data20 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data21 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data22 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data23 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data24 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data25 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data26 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data27 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data28 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data29 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data30 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data31 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data32 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
<tr>
<td>data33 Accession of NCBI: AF_06547, AF_06571, AF_07114, AF_08786, AF_06565, AF_12133, AKR0458</td>
</tr>
</tbody>
</table>