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 $O(n^3L^2)$, 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 [5, 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 $\sum$, where $\sum$ contains the 4 nucleotides {A, C, G, T} for nucleic acid sequences, or $\sum$ 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-89-2218-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:

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

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:

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

DM is one *Dayhoff* [7, 18] or *Blosum* [2] matrix.

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_e is related to the length of a gap. That is, the gap penalty is $P_g + kP_e$, called *affine gap penalty*, where P_g and P_e are both constants, $P_g \geq 0$, $P_e \geq 0$, 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

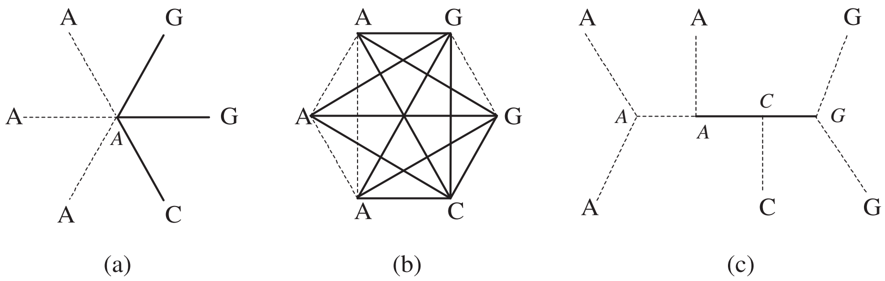

Figure 1: Alignment of six sequences A, A, A, G, G, C. (a) The star alignment, cost=3. (b) The SP (sum of pairs) alignment, cost=11. (c) The tree alignment, cost=2.

two input sequences are $a_1a_2\dots a_n$ and $b_1b_2\dots b_m$. Some notations are given as follows.

1. $A(i,j)$ is the score of an optimal alignment of $a_1a_2\dots a_i$ and $b_1b_2\dots b_j$.
2. $V(i,j)$ is the score of an optimal alignment of $a_1a_2\dots a_i$ and $b_1b_2\dots b_j$ whose last pair matches a_i with b_j.
3. $D(i,j)$ is the score of an optimal alignment of $a_1a_2\dots a_i$ and $b_1b_2\dots b_j$ whose last pair matches a_i with a space.
4. $I(i,j)$ is the score of an optimal alignment of $a_1a_2\dots a_i$ and $b_1b_2\dots b_j$ 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 ϵ.

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

Output: A multiple alignment of X and Y.

Notations: Let sequence X_k, $1 \le k \le m$, be denoted as $X_{k_1} X_{k_2} \cdots X_{k_{L_1}}$, and sequence Y_l, $1 \le l \le n$, be denoted as $Y_{l_1} Y_{l_2} \cdots Y_{l_{L_2}}$, 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 α denotes affine gap penalty, which can be adjusted.

Step 1: Compute the following:

$$D_{i,j} = \min \begin{cases} D_{i-1,j} + n\sum_{k=1}^{m} w(X_{ki}, -), \\ T_{i-1,j} + n\sum_{k=1}^{m} w(X_{ki}, -) + \alpha, \end{cases}$$

$$I_{i,j} = \min \begin{cases} I_{i,j-1} + m\sum_{l=1}^{n} w(-, Y_{lj}), \\ T_{i,j-1} + m\sum_{l=1}^{n} w(-, Y_{lj}) + \alpha, \end{cases}$$

$$T_{i,j} = \min \begin{cases} T_{i-1,j-1} + \sum_{k=1}^{m}\sum_{l=1}^{n} w(X_{ki}, Y_{lj}), \\ D_{i,j}, \\ I_{i,j}, \end{cases}$$

where $1 \le i \le L_1$, $1 \le j \le L_2$.

Step 2: After T_{L_1,L_2} 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) `AAGGCCTT`
(S_5) `-AGGGCTT`
(S_3) `-AGGGA-T`

Group Y : = (S_2) `-CGATT`
(S_4) `TCGA--`

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

Initial case:
$T_{0,0} = 0$.
$T_{i,j} = T_{i,j-1} + m\sum_{l=1}^{n} w(\texttt{-}, Y_{lj})$,
$i = 0,\ 1 \le j \le 6$.
$T_{i,j} = T_{i-1,j} + n\sum_{k=1}^{m} w(X_{ki}, \texttt{-})$,
$1 \le i \le 8,\ j = 0$.
Other case:

$$T_{i,j} = \min \begin{cases} T_{i-1,j-1} & + \sum_{k=1}^{m}\sum_{l=1}^{n} w(X_{ki}, Y_{lj}), \\ T_{i-1,j} & + n\sum_{k=1}^{m} w(X_{ki}, \texttt{-}), \\ T_{i,j-1} & + m\sum_{l=1}^{n} w(\texttt{-}, Y_{lj}), \end{cases}$$

where $1 \le i \le 8, 1 \le j \le 6$.

			-	-	C	G	A	T	T
			-	T	C	G	A	-	-
-	-	-	0	3	9	15	21	24	27
A	-	-	2	4	9	15	19	22	25
A	A	A	8	8	10	15	15	18	21
G	G	G	14	14	14	10	16	19	22
G	G	G	20	20	20	14	16	19	22
C	G	G	26	26	24	20	20	22	25
C	C	A	32	32	28	26	24	26	28
T	T	-	36	35	34	30	28	27	29
T	T	T	42	39	40	36	34	31	30

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

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

$T_{0,1} = T_{0,0} + 3w(\texttt{-},\texttt{-}) + 3w(\texttt{-},\texttt{T})$
$= 3$.
$T_{0,5} = T_{0,4} + 3w(\texttt{-},\texttt{T}) + 3w(\texttt{-},\texttt{-})$
$= 24$.
$T_{1,0} = T_{0,0} + 2w(\texttt{A},\texttt{-}) + 2w(\texttt{-},\texttt{-}) + 2w(\texttt{-},\texttt{-})$
$= 2$.
$T_{7,0} = T_{6,0} + 2w(\texttt{T},\texttt{-}) + 2w(\texttt{T},\texttt{-}) + 2w(\texttt{-},\texttt{-})$
$= 36$.

$$T_{1,1} = \min \begin{cases} T_{0,0} + w(\texttt{A},\texttt{-}) + w(\texttt{A},\texttt{T}) + \\ 2w(\texttt{-},\texttt{-}) + 2w(\texttt{-},\texttt{T}), \\ T_{0,1} + 2w(\texttt{A},\texttt{-}) + 4w(\texttt{-},\texttt{-}), \\ T_{1,0} + 3w(\texttt{-},\texttt{-}) + 3w(\texttt{-},\texttt{T}), \end{cases}$$
$$= 4.$$

$$T_{1,2} = \min \begin{cases} T_{0,1} + 2w(\texttt{A},\texttt{C}) + 4w(\texttt{-},\texttt{C}), \\ T_{0,2} + 2w(\texttt{A},\texttt{-}) + 4w(\texttt{-},\texttt{-}), \\ T_{1,1} + 3w(\texttt{-},\texttt{C}) + 3w(\texttt{-},\texttt{C}), \end{cases}$$
$$= 9.$$

$$T_{7,5} = \min \begin{cases} T_{6,4} + 2w(\texttt{T},\texttt{T}) + 2w(\texttt{T},\texttt{-})+ \\ w(\texttt{-},\texttt{T}) + w(\texttt{-},\texttt{-}), \\ \\ T_{6,5} + 2w(\texttt{T},\texttt{-}) + 2w(\texttt{T},\texttt{-})+ \\ 2w(\texttt{-},\texttt{-}), \\ \\ T_{7,4} + 3w(\texttt{-},\texttt{T}) + 3w(\texttt{-},\texttt{-}), \end{cases}$$
$$= 27.$$

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:

```
Group X              T
                     T
                     T
Group Y              T
                     -
```

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

```
Group X              CTT
                     CTT
                     A-T
Group Y              ATT
                     A--
```

Finally, backtracking from $T_{1,0}$ to $T_{0,0}$ indicates the following alignment:

```
Group X              AAGGCCTT
                     -AGGGCTT
                     -AGGGA-T
Group Y              --CG-ATT
                     -TCG-A--
```

The final multiple sequence alignment is given as follows:

$S_1 = \texttt{AAGGCCTT}$
$S_2 = \texttt{--CG-ATT}$
$S_3 = \texttt{-AGGGA-T}$
$S_4 = \texttt{-TCG-A--}$
$S_5 = \texttt{-AGGGCTT}$

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 $L = max\{|S_1|,\ |S_2|,\ \cdots,\ |S_n|\}$, 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.

	S_1	S_2	S_3	S_4	S_5
S_1	-	5	4	7	2
S_2		-	4	3	4
S_3			-	4	2
S_4				-	6
S_5					-

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.

$$S_1 = \texttt{AAGGCCTT}$$
$$S_2 = \texttt{CGATT}$$
$$S_3 = \texttt{AGGGAT}$$
$$S_4 = \texttt{TCGA}$$
$$S_5 = \texttt{AGGGCTT}$$

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 7. 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 $G_1 = \{S_1, S_5, S_3\}$. 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 4. 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:

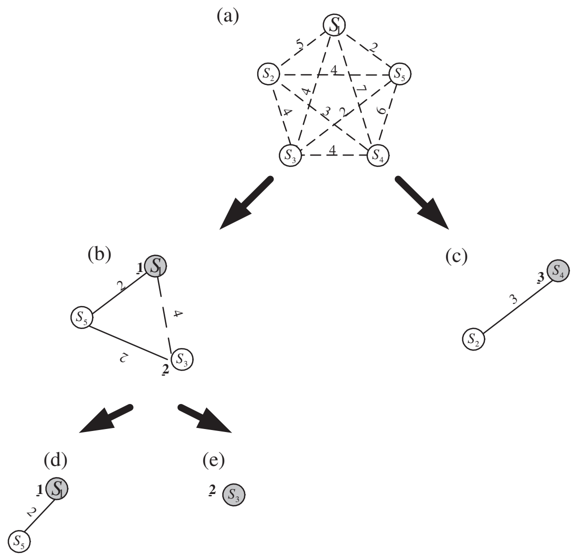

Figure 3: The clustering method.

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.

	A	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	Y
A	2																			
C	-2	12																		
D	0	-5	4																	
E	0	-5	3	4																
F	-3	-4	-6	-5	9															
G	1	-3	1	0	-5	5														
H	-1	-3	1	1	-2	-2	6													
I	-1	-2	-2	-2	1	-3	-2	5												
K	-1	-5	0	0	-5	-2	0	-2	5											
L	-2	-6	-4	-3	2	-4	-2	2	-3	6										
M	-1	-5	-3	-2	0	-3	-2	2	0	4	6									
N	0	-4	2	1	-3	0	2	-2	1	-3	-2	2								
P	1	-3	-1	-1	-5	0	0	-2	-1	-3	-2	0	6							
Q	0	-5	2	2	-5	-1	3	-2	1	-2	-1	1	0	4						
R	-2	-4	-1	-1	-4	-3	2	-2	3	-3	0	0	0	1	6					
S	1	0	0	0	-3	1	-1	-1	0	-3	-2	1	1	-1	0	2				
T	1	-2	0	0	-3	0	-1	0	0	-2	-1	0	0	-1	-1	1	3			
V	0	-2	-2	-2	-1	-1	-2	4	-2	2	2	-2	-1	-2	-2	-1	0	4		
W	-6	-8	-7	-7	0	-7	-3	-5	-3	-2	-4	-4	-6	-5	2	-2	-5	-6	17	
Y	-3	0	-4	-4	7	-5	0	-1	-4	-1	-2	-2	-5	-4	-4	-3	-3	-2	0	10

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

Table 3: The score and time of our algorithm, mixed algorithm (NJ + our group alignment method) and Clustal W algorithm with various parameters on real biological sequences.

			Our algorithm		Mixed algorithm		Clustal W	
test data	M	C	score	time	score	time	score	time
data1			17311	0.101265	17342	0.107230	17324	0.173129
data2			222826	1.187414	222826	1.138993	222826	1.451267
data3			68540	0.324586	71012	0.333779	68887	0.424164
data4			43004	0.259825	43098	0.266993	43004	0.399287
data5			59439	0.349632	59439	0.354719	59439	0.502054
data6			42433	0.252039	42433	0.254458	42433	0.375801
data7			264897	1.344202	265199	1.350774	264897	1.627989
data8			40258	0.312156	40320	0.316710	40278	0.482699
data9			91901	0.671743	92182	0.683620	91913	0.978295
data10			41375	0.342158	41375	0.354062	41375	0.555916
data11			26066	0.228225	26094	0.230667	26077	0.376905
data12			148811	0.955186	149653	0.984842	149166	1.308886
data13	v		22973	0.200235	22963	0.201590	23012	0.333149
data14			24202	0.229099	24281	0.223883	24202	0.379132
data15	v	v	272354	1.960959	272288	1.975593	272245	2.490145
data16	v	v	86513	0.684733	86479	0.676518	86479	0.980272
data17			111216	0.868870	111216	0.843113	111216	1.205047
data18			123890	0.880375	125101	0.907264	123955	1.213583
data19		v	162194	1.027529	166969	1.050819	162122	1.386224
data20			13488	0.064967	13565	0.066114	13508	0.107731
data21			80050	0.510627	80181	0.506351	80164	0.729023
data22			15409	0.013634	15626	0.010781	15412	0.017356
data23			316212	1.445035	317329	1.514042	316212	1.876586
data24			99983	0.122478	100682	0.113243	100223	0.166836
data25		v	500229	4.113774	500270	4.171516	499949	5.052194
data26			263229	0.901181	263674	0.929714	264180	1.103604
data27			329031	2.015295	329031	1.948144	329031	2.424556
data28			32929	0.128120	32952	0.128251	33000	0.192936
data29			66511	0.220290	66511	0.220217	66511	0.315244
data30			40715	0.150231	40715	0.150305	40715	0.221670
data31			507331	3.432342	508095	3.436299	507356	4.319445
data32			158259	0.415546	160615	0.419811	158259	0.521249
data33			636279	5.860559	636376	5.872305	636279	7.310256

In column M, if one entry is marked, it means that our result is worse than that of the mixed algorithm. Similarly, column C is used for the comparison of our algorithm and Clustal W.

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 program to verify the algorithm and we do not find any counter test case until now. We are now trying to prove its correctness.

References

[1] S. F. Altschul. Gap costs for multiple sequence alignment. *Journal of Theoretical Biology*, 138:297–309, 1989.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[3] S. F. Altschul and D. J. Lipman. Trees, stars and multiple sequence alignment. *SIAM Journal on Applied Mathematics*, 49(1):197–209, 1989.

[4] V. Bafna, E. L. Lawler, and P. Pevzner. Approximation algorithms for multiple sequence alignment. In *In 5th Ann. Symp. On Pattern Combinatorial Matching*, volume 807, pages 43–53, 1994.

[5] H. Carrillo and D. J. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48:1073–1082, 1988.

[6] S. C. Chan, A. K. C. Wong, and D. K. Y. Chiu. A survey of multiple sequence comparison methods. *Bulletin of Mathematical Biology*, 54:563–598, 1992.

[7] M. O. Dayhoff. *Atlas of Protein Sequence and Structure.* National Biomedical Research Foundation, Washington DC, 1978.

[8] O. Gotoh. Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Computer Applications in the Biosciences*, 9:361–370, 1993.

[9] S. K. Gupta, J. D. Kececioglu, and A. A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2(3):459–472, 1995.

[10] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 30:141–154, 1993.

[11] Chia Mao Huang and Chang Biau Yang. Approximation algorithms for constructing evolutionary trees. In *Proc. of National Computer Symposium, Workshop on Algorithm and Computation Theory*, pages A099–A109, 2001.

[12] T. Jiang, E. L. Lawler, and L. Wang. Aligning sequences via an evolutionary tree: Complexity and approximation. In *In Proceedings of the Symposium on the Theoretical Aspects of Computer Science*, pages 760–769, 1994.

[13] T. Jiang and L. Wang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.

[14] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *In 4th Ann. Symp. On Pattern Combinatorial Matching, Springer Verlag Lecture notes in Computer Science*, volume 684, pages 106–119, 1993.

[15] D. J. Lipman, S. F. Altschul, and J. D. Kececioglu. A tool for multiple sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America*, 86:4412–4415, 1989.

[16] P. A. Pevsner. Multiple alignment, communication cost and graph matching. *SIAM Journal on Applied Mathematics*, 52:1763–1779, 1992.

[17] N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4:406–425, 1987.

[18] R. M. Schwartz and M. O. Dayhoff. *Matrices for detecting distant relationships.* In M. Dayhoff, editor, Atlas of Protein Sequence and Structure, volume 5, pages 353-358. National Biomedical Research Foundation, Washington, DC, 1979.

[19] J. Stoye, S. W. Perrey, and A. W. M. Dress. Improving the divide-and-conquer approach to sum-of-pairs multiple sequence alignment. *Applied Mathematics Letters*, 10(2):67–73, 1997.

[20] W. R. Taylor. Multiple sequence alignment by a pairwise algorithm. *Computer Applications in the Biosciences*, 3:81–87, 1987.

[21] J. D. Thompson, D. G. Higgins, and T. J. Gibson. Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, positions-specific gap penalties and weight matrix choice. *Nucleic acids res*, 22(22):4673–4680, 1994.

[22] U. Tonges, S. W. Perrey, J. Stoye, and A. W. M. Dress. A general method for fast multiple sequence alignment. *Gene*, 172(1):33–41, 1996.

Table 4: The source of real biological test data.

	source of test data
data1	Accession of NCBI: NP_066226, NP_066206, NP_066204, NP_065431
data2	Accession of NCBI: NP_066558, BAB20723, NP_112135, NP_112096, NP_112109, NP_007565, NP_112122
	NP_110507, BAB40351, BAB20736, AAF61381, BAB20710, NP_112525,
data3	Accession of NCBI: NP_066226, NP_066206, NP_066204, NP_065431, NP_112525, NP_007565, NP_112122
data4	Accession of NCBI: NP_008345, NP_008215, NP_008228, BAA95619, AAK38692, AAB00992
data5	Accession of NCBI: AAK08554, AAK08578, NP_071645, NP_068785, NP_007371, NP_008098, NP_007384
data6	Accession of NCBI: AAK14328, AAK14327, AAG49582, NP_008649, NP_008279, AAK00949
data7	Accession of NCBI: NP_112525, NP_007565, NP_112122, NP_007371, NP_008098, AAB00992, AAK38692
	NP_007384, NP_008345, NP_008215, NP_008228, NP_008046, AAD34188, BAA95619
data8	Accession of NCBI: NP_066223, AAG60030, NP_066216, NP_066201, NP_065428
data9	Accession of NCBI: AAK53588, NP_059474, NP_008289, NP_008659, NP_008342, NP_008225, NP_008212
data10	Accession of NCBI: NP_112522, NP_007562, NP_008342, NP_008225, NP_008212
data11	Accession of NCBI: NP_00824, NP_11433, AAK38689, CAC37083
data12	Accession of NCBI: NP_06622, AAG60030, NP_06621, NP_06620, NP_06542, NP_06396, NP_03825,
	NP_11437, AAK51683
data13	Accession of NCBI: NP_06396, NP_03825, NP_11437, AAK51683
data14	Accession of NCBI: AAK08547, AAK08571, NP_071642, NP_068782
data15	Accession of NCBI: NP_066555, NP_112132, BAB40348, AAF61378, BAB20733, BAB20720, BAB20707,
	NP_110504, NP_114224, AAK52942, NP_112522, NP_007562
data16	Accession of NCBI: AAK08547, AAK08571, NP_07164, NP_06878, NP_06655, NP_11213, BAB40348
data17	Accession of NCBI: NP_008342, NP_008225, NP_008212, NP_007381, NP_007368, NP_008095,
	AAD34185, NP_008043
data18	Accession of NCBI: AAK08547, AAK08571, NP_071642, NP_068782, AAK53588, NP_059474, NP_008289,
	NP_008659
data19	Accession of NCBI: NP_066223, AAG60030, NP_066216, NP_066201, NP_065428, AAK53588, NP_059474,
	NP_008289, NP_008659
data20	Accession of NCBI: LGHO2, S33878, S14719, S11538
data21	Accession of NCBI: P14315, P47757, P47756, P34686, P13021, P48603, P13517
data22	Accession of NCBI: G29501, H29501, E29501, B29501, C29501, A32654, A29501, F29501, I29501,
	C28854, B28854, A28854
data23	Accession of NCBI: NP_007567, CAC36948, NP_007373, NP_008100, NP_007386, NP_008217, NP_008230,
	NP_008048, AAG28221, BAA95621, AAK38694, NP_115355, NP_115433, NP_071647, NP_068787
data24	Accession of NCBI: NP_007566, CAC36947, NP_007372, NP_008099, NP_007385, NP_008216, NP_007839,
	NP_008047, AAG28220, BAA95620, AAK38693, NP_115354, NP_115432, NP_071646, NP_068786
data25	Accession of NCBI: NP_007574, NP_112532, NP_007380, NP_008107, NP_007393, BAA85284, NP_008237,
	NP_008055, AAG28215, BAA95628, AAK38699, NP_115361, NP_115439, NP_071654, NP_068794
data26	Accession of NCBI: NP_007573, NP_112531, NP_007379, NP_008106, NP_007392, NP_008223, NP_008236,
	NP_008054, AAG28227, BAA95627, AAK38700, NP_115362, NP_115440, NP_071653, NP_068793
data27	Accession of NCBI: NP_007568, NP_112528, NP_007374, NP_008101, NP_007387, BAA85281, NP_008231,
	NP_008049, AAG28222, BAA95622, AAK38695, NP_115356, NP_115434, NP_071648, NP_068788
data28	Accession of NCBI: LGHOD, LGHO, S33875, S33877, A45542, A05144
data29	Accession of NCBI: HAAQ, HACQ, A26543, HAHOD, P01923, P06635, HAOR ,P01965, A45964
data30	Accession of NCBI: HBAQ, HBCQ, B26543, HBGO, 223012, HBOR, P02067
data31	Accession of NCBI: NP_007563, NP_112523, NP_007369, NP_008096, NP_007382, BAA85278, NP_008226,
	NP_00804, AAG28217, BAA95617, AAK38690, NP_115351, NP_115429, NP_071643, NP_068783
data32	Accession of NCBI: NP_007569, NP_112727, NP_007375, NP_008102, NP_007388, BAA85282, NP_008232,
	NP_008050, AAG28223, BAA95623, AAK38696, NP_115357, NP_115435, NP_071649, NP_068789
data33	Accession of NCBI: NP_007571, NP_112728, NP_007377, NP_008104, NP_007390, BAA85283, NP_008234,
	NP_008052, AAG28225, BAA95625, AAK38697, NP_115359, NP_115437, NP_071651, NP_068791