# 클러스터링 기반 다중 서열 정렬 알고리즘

이병일*, 이종연**, 정순기*
*충북대학교 컴퓨터공학과, **충북대학교 컴퓨터교육과
e-mail: lbi5320@cbnu.ac.kr

# Algorithm of Clustering-based Multiple Sequence Alignment

Byung-Il*, Jong-Yun Lee**, and Soon-Key Jung*
*Dept. of Computer Engineering, Chungbuk National University
**Dept. of Computer Education, Chungbuk National University

요 약

3개 이상의 DNA 혹은 단백질의 염기서열을 정렬하는 다중 서열 정렬(multiple sequence alignment, MSA)은 서열들 사이의 진화관계, 단백질의 구조와 기능에 관한 연구에 필수적인 도구이다. 최적화된 다중서열 정렬을 얻기 위해 사용되는 가장 유용한 방법은 동적 프로그래밍이다. 그러나 동적프로그래밍은 정렬하고자 하는 서열의 수가 증가함에 따라 시간도 지수함수($O(n^k)$)로 증가하기 때문에 다중 서열 정렬에는 효율적이지 못하다. 따라서, 본 논문에서는 최적의 MSA 문제를 해결하기 위해 클러스터링 기반의 새로운 다중 서열 정렬 (Clustering-based Multiple Sequence Alignment, CMSA) 알고리즘을 제안한다. 결과적으로 제안한 CMSA 알고리즘의 기여도는 다중 서열 정렬의 질적 향상과 처리 시간 단축($O(n^3L^2)$)이 기대된다.

## 1. 서론

다중 서열 정렬(MSA)은 지난 10년동안 단백질과 핵산의 서열들을 분석하는데 주요한 도구가 되어왔으며, 생물학적 서열들의 기능적, 구조적 그리고 진화적인 부분을 연구하는데 매우 중요하다[1]. 이러한 중요성 때문에 많은 알고리즘들이 제안되어 왔다. 다중 서열 정렬 문제는 주어진 임의의 표준 점수에 기초하여 가장 좋은 점수를 가지고 일련의 서열들로 부터 정렬을 구하는 것이다. 생물학적 서열은 4개의 염기나 20개의 아미노산으로 구성된 일련의 문자들의 합으로 이루어져 있으며, 공백은 '-'으로 표현된다.

다중 서열 정렬의 기본적인 방법은 모든 서열들 중에서 유사성을 최대화하거나, 거리를 최소화하기 위해 전역 정렬로 구성된다. 동적 프로그래밍(dynamic programming)은 정렬 문제 중 가장 잘 알려진 방법으로 한 쌍의 서열에서 동적 프로그래밍을 사용하여 최적의 정렬을 구할 때 $O(n^2)$의 처리 시간이 걸린다. 하지만 서열의 수가 증가함에 따라 시간도 지수적으로 증가하는 문제점을 가지고 있다. 즉, $k$개의 서열들을 정렬하는 일반적인 문제에서는 $O(n^k)$의 시간이 요구된다[2].

지금까지 다중 서열 정렬의 우수한 측정을 위해 많은 기준들이 제시되었는데, 그 중 가장 많이 사용되는 기준은 sum-of-pairs이다[1]. 이것은 Dayhoff나 Blosum행렬로 각각의 쌍 정렬의 점수를 산출하고, 쌍 정렬들의 모든 점수 합을 가지고 다중서열 정렬을 위한 점수를 생성한다[3].

따라서, 본 논문에서는 다중 서열정렬에 이진트리 형태의 클러스터링 기반 다중 서열 정렬(CMSA) 알고리즘을 제안한다. 제안한 정렬 알고리즘은, 긴 길이를 가진 두 서열들을 두 개의 클러스터로 나누는 것에 기본 개념을 기반하며 시간 복잡도는 $O(n^3L^2)$이다. 실험결과, Clustal W 보다 정렬의 질과 실행 시간의 개선이 기대된다.

* 본 연구는 2005년도 과기부 지방연구중심대학육성지원 사업 (제1세부과제)에 의해 수행되었음.

## 2. 관련 연구

### 2.1 다중 서열 정렬

다중 서열 정렬에는 크게 2가지로 나눌 수 있다. 첫째, 전역 정렬은 서열 전체를 정렬하는 방법으로 유사성을 극대화시키기 위한 정렬 방법이다. 둘째, 지역 정렬은 모티프, 또는 homologous 부분 중 일치되는 부분들을 찾는 방법이다. 본 논문에서는 단지 전역 정렬만을 고려한다.

다중 서열 정렬은 일련의 서열들에 gap 문자'-'을 각각의 서열들에 삽입하여 얻을 수 있다. 그러므로 모든 결과 서열들의 길이는 같고, 오직 gap 문자만으로 구성된 열은 존재하지 않는다. 일반적인 다중 서열 정렬은 다음과 같다.

**정의1**: 문자열 $S_1, S_2, \ldots, S_k$이 주어지고, 공백을 포함한 문자열 $A_1, A_2, \ldots, A_k$에서 다중 서열 정렬을 만들때

1. $|A_1| = |A_2| = \ldots = |A_k|$
2. $A_i$ 에서 모든 gap 문자 '-'을 제거하면 $S_i$와 같다.
3. 오직 gap 문자만 가진 열은 존재하지 않는다.

가장 좋은 정렬을 가지고 정렬을 찾고자 할 때, MSA의 점수는 모든 열들의 점수 합을 말한다. 여기에는 두 가지 종류의 입력 서열 데이터가 있다. 즉, DNA 서열과 단백질 서열이다. 두 DNA 서열에서 가장 간단한 비용 함수는 다음과 같은 편집 거리를 따른다.

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

여기서, $c_1$ 은 일치, $c_2$ 는 치환을 의미하며, $c_3$은 affine gap cost를 의미한다.

단백질 서열에서, 치환 행렬($TM$)은 다름과 같이 사용한다.

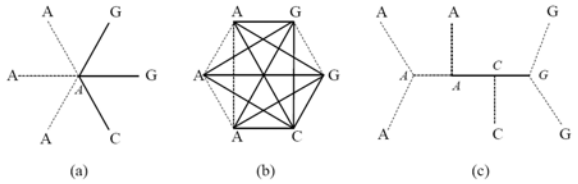
$$\delta(x,y)=\begin{cases} TM(x,y) \ \ if \ \ x\neq gap \ \ and \ \ y\neq gap, \\ 0 \ \ if \ \ x=gap \ \ and \ \ y=gap, \\ affine \ \ gap \ \ cost \ \ if \ \ x=gap \ \ or \ \ y=gap. \end{cases}$$

여기서, $DM$은 Dayhoff나 Blosum 행렬을 말한다.

다중 서열 정렬에서는 다양한 결과들을 만족시키기 위한 목적으로 여러 가지의 점수 행렬을 사용한다. 일반적으로, 최적화 문제로서 다중 서열 정렬 문제를 표현할 수 있다. 각 열을 평가하기 위해 sum-of-pair(SP) 측정, 트리 정렬, 성형(star) 정렬등 세 개의 행렬들이 사용된다[4]. 스타 정렬에서, 점수는 모든 서열들과 선택된 일치 서열 사이의 쌍 정렬의 점수의 합이다. 트리 정렬에서, 각 트리의 모서리의 쌍 점수를 요약함으로서 진화적인 트리에 점수를 평가하는 것이 가능하다. 이것은 두 노드 사이의 진화적인 거리를 표현한다. 그러므로 이러한 점수 방법은 진화적인 거리를 최소화함으로서 진화 과정을 기술한다.

SP 측정의 일반적인 정의는 다음과 같다.

**정의2** : $k$의 서열들 중 다중 서열 정렬 A에 대한 sum-of-pairs는 A의 모든 $\binom{k}{2}$ 쌍 정렬 점수의 합이다.




(그림 1) 6개 서열들 A, A, A, G, G, C의 정렬: (a)스타 정렬; (b)SP 정렬; (c)트리 정렬

(그림1)은 위의 점수 규칙의 예를 보여준다. 만약 하나의 열에 두 개의 동일한 문자들이 있다면 0의 값을 받고, 그렇지 않다면 1의 값을 받는다. (그림1a)는 스타 정렬로 3, (그림1b)는 SP(sum of pairs) 정렬로 11, (그림1c)는 트리 정렬로 2의 값을 가진다.

### 2.2 Affined gap penalty

갭의 벌점에는 두가지가 있다. 벌점 $P_g$는 갭의 시작과 관계된다. 또 다른 벌점 $P_e$는 갭의 길이와 관계된다. 따라서, 갭 벌점은 $P_g+kP_e$이다. 이것을 affined gap penalty라고 부른다. 이 때 $P_g$와 $P_e$는 상수이고, $P_g\geq 0$, $P_e\geq 0$, 그리고 $k\geq 1$은 갭의 길이 이다. Affine gap penalty를 가지고 정렬을 찾는 문제는 동적 프로그래밍 접근으로 풀 수 있다.

### 2.3 Clustal W를 이용한 다중 정렬

다중 서열 정렬을 위해 많이 사용하는 프로그램으로 Clustal W가 있다[5]. Clustal W에서 사용되는 휴리스틱은 계통 발생학적 분석에 기반 한 것이다. 먼저 정렬할 모든 서열들에 대한 짝짓기 거리 행렬(pairwise distance matrix)를 만들고, 이웃 결합(neighbor-joining) 알고리즘을 이용하여 유도 트리(guide tree)를 작성한다. 그리고 서열 가운데 가장 관계가 가까운 짝 즉 트리의 가장 바깥쪽 가지를 동적 프로그래밍으로 정렬한다. 그 다음 각각의 새로운 정렬을 분석하여 서열의 프로필을 작성한다. 마지막으로(트리의 구조에 따라서) 전체 정렬이 이루어 질 때까지 각 정렬 프로필을 서로 정렬시키거나 다른 서열과 정렬 한다.

단백질 서열 정렬에 있어서 Clustal W의 휴리스틱 전략 가운데 하나는 각 정렬을 수행할 때 예산 진화 거리에 기반한 다른 측정 행렬을 사용한다. 두 서열이 트리에서 가까운 거리에 있으면 근연관계의 정렬에 최적화된 측정 행렬을 사용하고, 멀리 떨어져 있는 서열들의 경우에는 원연관계에 최적화된 측정 행렬을 사용한다. 따라서 모든 짝 정렬에 동일한 측정 행렬을 사용하기 보다는 가까운 관계에는 BLOSUM62를 선택하고 좀더 먼 관계일 경우에는 BLOSUM45를 사용한다.

## 3. 그룹 정렬 방법

이 장에서는 그룹 정렬 방법과, 알고리즘의 동작 원리에 대하여 기술한다. 각각 정렬되어 있는 두개의 서열이 주어졌을 때, 그룹 정렬 방법은 두개의 서열을 하나의 서열로 결합하여 모든 서열들을 정렬한다. 그룹 정렬의 기본적인 알고리즘은 (그림2)와 같다.

**Input:** Two alignment $X=\{X_1, X_2, ..., X_m\}$ and $Y=\{Y_1, Y_2, ..., Y_n\}$, where each $X_k$, $1 \leq k \leq$ m, or $Y_l$, $1 \leq l \leq$ n.

**Output:** A multiple alignment of $X$ and $Y$

1. 다음과 같이 계산

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

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

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

여기서, $1 \leq i \leq 8, 1 \leq j \leq 6$ 이다.

2. $T_{L1,L2}$을 찾은 후, $X$ 와 $Y$의 다중 서열 정렬을 찾기 위해 역으로 거슬러 올라간다.

(그림 2) 그룹 정렬 알고리즘

위 그룹 정렬에서 시간 복잡도는 $O(nmL_1L_2)$이다. 서열 $S_k$와 일련의 서열들 G 사이의 거리는 $d(S_k, G)$로 표기한다. 이것은 $S_k$와 $G$의 모든 서열 사이에서 가장 작은 거리를 정의한다.

위의 그룹 정렬 방법을 가지고 그룹 $X$와 그룹 $Y$의 최종 다중 서열 정렬을 만든다. 점수 함수에서 $\alpha$=0, 일치는 0, 불일치는 1, 삽입/삭제는 1의 값을 가진다.

Group X := ($S_1$) AAGGCCTT
($S_2$) -AGGGCTT
($S_3$) -AGGGA-T

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

여기서, $m$=3, $n$=2, $L_1$=8, $L_2$=6 이다.
그룹 정렬의 계산식은 다음과 같다.

**Initial case:**

$$T_{0,0} = 0,$$

$$T_{i,j} = T_{i,j} + m\sum_{l=1}^{n} w(-, Y_{lj}),$$

$$i = 0, 1 \le j \le 6.$$

$$T_{i,j} = T_{i-1,j} + n\sum_{k=1}^{m} w(X_{ki}, -),$$

$$1 \le i \le 8, j = 0.$$

**Other case:**

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

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

그룹 정렬을 통한 계산 결과는 (그림3)에서 보여준다.

| | | | - | - | 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 | 16 | 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 |

(그림 3) 그룹 X(2) 와 그룹 Y(3)의 그룹 정렬 방법

(그림3)에서 정렬을 얻기 위해 오른쪽 끝에서부터 역으로 검색하였다. 위쪽 화살표는 그룹 Y의 모든 서열에 gap 삽입을 의미하고, 왼쪽 화살표는 모든 서열 그룹 X에 gap을 삽입한다는 의미이다. 최종 다중 서열 정렬은 다음과 같다.

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

## 4. 제안하는 다중 서열 정렬 알고리즘

앞 장에서 그룹 정렬 방법에 대해 논하였다. 이 장에서는 CMSA 알고리즘을 제안한다. 제안한 CMSA 알고리즘(그림4)의 기본적은 생각은 각 그룹 안에 있는 gap의 수를 줄이는 것이다. 그러므로 만약 서로 다른 두개의 그룹 안 에서 거리가 가장 긴 두개의 서열들이 제공될 때, 입력되는 서열들이 매우 유사하다면 좀더 좋은 다중 서열 정렬을 구할 수 있다. 제안된 알고리즘에 대한 자세한 기술은 (그림4)와 같다.

입력 : 일련의 서열들 $S = \{S_1, S_2, \ldots, S_n\}$
출력 : $S$ 의 다중 서열 정렬
1. If |S| ≤ 1, then stop.
2. Construct the distance matrix for S
3. Sort all entries in the distance matrix
4. Create a set of sequences R=S.
5. Select a pair of sequences $S_i$ and $S_j$
6. Let $G_1$={$S_1$} and $G_2$={$S_2$}, $R = R - \{S_i, S_j\}$
   6.1 Select $S_k \in R$
   6.2 If $d(S_k, G_1) \le d(S_k, G_2)$, then $G_1 = G_1 \cup \{S_k\}$;
   otherwise $G_2 = G_2 \cup \{S_k\}$
   6.3 $R = R - \{S_k\}$
7. Recursively apply CMSA by setting the input $S = G_1$
   Recursively apply CMSA by setting the input $S = G_2$
8. Perform group alignment method on $G_1$ and $G_2$

(그림 4) CMSA 알고리즘

단계2에서 S의 각 쌍의 서열들의 최적 정렬을 계산하고 나서, S의 거리 행렬을 생성한다. 단계3에서는 순서의 증감 없이 거리 행렬에 의해 분류하고, 단계5에서는 가장 긴 거리를 가진 $S_i$와 $S_j$ 서열을 선택한다. 단계6에서는 R이 공백이 될 때까지 단계6.1~단계6.3을 실행한다. 단계7에서 입력이 S=$G_1$와 S=$G_2$이 될 때까지 반복하여 CMSA 알고리즘을 적용한다. 단계8에서 $G_1$와 $G_2$에서 그룹 정렬 방법을 수행한다.

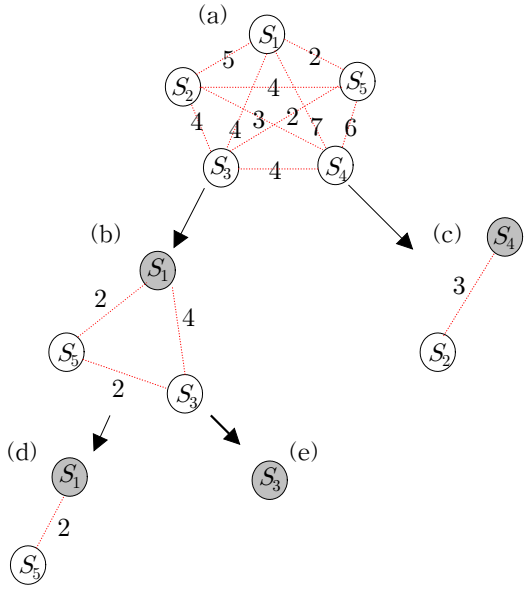
제안한 알고리즘을 단계별로 자세히 설명하면 다음과 같다. 예로, 5개의 서열들을 가지고 생각할 때, $\alpha$=0이고 일치는 0, 불일치와 삽입/삭제는 1이라고 가정한다.

단계2 에서 얻은 거리 행렬을 <표1>에 나타낸다.

<표 1> 5 서열들의 거리 행렬

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

단계5와 단계6에서 서열들을 두개의 그룹으로 나눈다. 나누는 과정은 (그림5)에서 보여 주고 있다.
각 그룹에 놓인 첫번째 서열은 회색 노드로 표현한다.각 노드에 연결된 수는 서열이 추가하는 순서를 나타낸다. 실선은 서열과 그 그룹사이의 거리를 나타낸다. 그림 5(b)와




(그림 5) 클러스터링 방법

(c)는 그림 5(a)로부터 분리된 것이다. 처음에, $S_1$와 $S_4$는 긴 거리 7을 가지기 때문에 $G_1$={$S_1$}와 $G_2$={$S_4$}이다. $S_5$는$G_1$에 가깝기 때문에, $S_5$를 $G_1$에 추가하면, $G_1$={$S_1, S_5$}가 된다. $S_3$ 또한 $S_5$와 가깝기 때문에 $S_3$도 $G_1$에 추가되어, $G_1$={$S_1, S_5, S_3$}이 된다. 마지막으로, $S_2$은 $S_4$와 가깝기 때문에 $S_2$를 $G_2$에 추가시킨다. 마지막 클러스터링 결과는 $G_1$={$S_1$, $S_3$, $S_5$} 와 $G_2$={$S_2$, $S_4$} 이다.

그룹 $G_1$의 서열들의 수가 2보다 크기 때문에, $G_1$을 다시 분할하는 것을 (그림5(d)) 와 (e)에서 보여주고 있다. (그림5(d))와 (e)에서, $S_1$와 $S_3$가 가장 긴 거리 4를 가지고 있으므로, G′$_1$ = {$S_1$}와 G′$_2$ = {$S_3$}로 나눈다. $S_5$는 G′$_1$와 가깝기 때문에 G′$_1$ 에 추가하면 G′$_1$ ={$S_1, S_5$}이 된다. 그러면 3개의 그룹 G′$_1$ ={$S_1, S_5$}, G′$_2$ ={$S_3$}, $G_2$={$S_2, S_4$}으로 구성된다.

## 5. 실험 결과

이 장에서는 실험 결과와 제안한 알고리즘의 실행을 분석하였다. 모든 실험은 LINUX 운영체제의 메모리 512M, Pentium4 PC에서 실행하였다. 테스트에 사용된 서열들은 NCBI에 있는 실제 생물학적 서열들이다<표2>. 다중 서열 정렬의 우수함을 결정하기 위해 sum-of-pair 측정을 사용하였고, 여기에 사용된 점수 행렬은 PAM250이다.

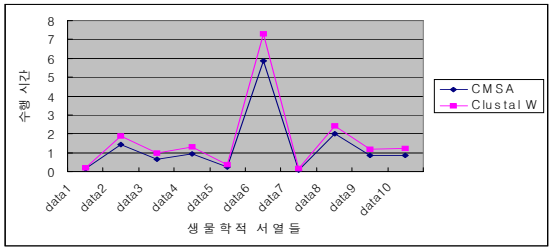
<표 2> 실제 생물학 테스트 데이터

| | |
|---|---|
| data1 | HBAQ, HACQ, B26543, HBGO, 23012, HBOR, P02067 |
| data2 | NP_007567, CAC36948, NP_007373, NP_008100, NP_007386 |
| data3 | AAK53588, NP_059474, NP_008289, NP_008659, NP_008342 |
| data4 | NP_06622, AAG60030, NP_06621, NP_06620, NP_06542 |
| data5 | AAK14328, AAK14327, AAG49582, NP_008649, NP_008279 |
| data6 | NP_007571, NP_112728, NP_007377, NP_008104, NP_007390 |
| data7 | NP_066226, NP066204, NP_066204, NP_065431 |
| data8 | NP_007568, NP_112528, NP_007374, NP_008108, NP_007387 |
| data9 | AAK08547, AAK08571, NP_07164, NP_06878, NP_06655 |
| data10 | NP_008342, NP_008225, NP_008212, NP_007381, NP_007368 |

본 논문에서는 클러스터링 방법을 이용한 그룹 정렬 방법을 제안하였다. 제안한 알고리즘(CMSA)과 Clustal W의 성능 평가는 <표3>와 같고, 두 알고리즘의 수행시간 비교는 (그림6)과 같다. 실험 결과, 제시한 알고리즘이 Clustal W 보다 빠르고, 점수도 향상됨을 알 수 있다

<표 2> CMSA와 Clustal W의 비교

| | CMSA | | Clustal W | |
|---|---|---|---|---|
| test data | score | time | score | time |
| data1 | 40715 | 0.150231 | 40715 | 0.221670 |
| data2 | 316212 | 1.445035 | 316212 | 1.876586 |
| data3 | 91901 | 0.671743 | 91913 | 0.978295 |
| data4 | 148811 | 0.955186 | 149166 | 1.308886 |
| data5 | 42433 | 0.252039 | 42433 | 0.375801 |
| data6 | 636279 | 5.860559 | 636279 | 7.310256 |
| data7 | 17311 | 0.101265 | 17324 | 0.173129 |
| data8 | 329031 | 2.015295 | 329031 | 2.424556 |
| data9 | 111216 | 0.868870 | 111216 | 1.205047 |
| data10 | 123890 | 0.880375 | 123955 | 1.213583 |




(그림 6) CMSA와 Clustal W의 시간 비교

## 6. 결론

본 논문에서는 다중 서열 정렬 문제를 풀기 위해 CMSA 알고리즘을 제안하였다. 이것은 클러스터링 방법과 그룹 정렬 방법으로 이루어진 것이다. 동적 프로그래밍을 이용한 쌍 정렬에서의 최적 정렬을 찾을때 $O(n^2)$의 시간이 소요된다. 이러한 개념을 기반으로 최적의 MSA 문제를 해결하는데 $O(n^3L^2)$의 시간이 요구되는 CMSA 알고리즘을 제안하였다. 알고리즘을 검증하기 위해 프로그램을 기술하였고, 실험 결과를 통해 다중 서열 정렬의 질과 처리 속도가 향상됨을 보여주었다. 향후 이 알고리즘의 정확성을 증명하기 위해 여러 가지 다중 서열 정렬 프로그램들과의 성능 비교도 필요하다.

## 참고문헌

[1] Y, Wang and K, Li, "An adaptive and iterative algorithm for refining multiple sequence alignment," *Computational Biology and Chemistry* 28, pages 141-148, 2004

[2] J. Kececioglu, "The maximum weight trace problem in multiple sequence alignment," In *In 4th Ann. Symp. on Pattern Combinatorial Matching,* volume 684, pages 106-119, 1993.

[4] J. Heringa, "Local weighting schemes for protein multiple sequence alignment," Computers and Chemistry 26, pages 459-477, 2002

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