韓國 컴퓨터情報學會 論文誌
第10卷 第5號, 2005. 11.

2005-10-5-1-1

클러스터링 분기를 이용한 다중 서열 정렬 알고리즘

이 병 일*, 이 종 연**, 정 순 기***

A Multiple Sequence Alignment Algorithm using Clustering Divergence

Byung-Il Lee*, Jong-Yun Lee**, Soon-Key Jung***

요 약

다중 서열 정렬(multiple sequence alignment, MSA)은 단백질과 핵산 서열들의 분석에 필요한 가장 중요한 도구이다. 생물학적인 서열들은 그들 사이의 유사성과 차이점을 보여주기 위하여 각각의 서열들을 수직적으로 정렬한다. 본 논문에서는 클러스터링 분기를 이용하여 두 그룹의 서열들 사이에서 정렬을 수행하는 효율적인 그룹 정렬 방법을 제안하였다. 제안한 알고리즘(Multiple Sequence Alignment using Clustering Divergence : CDMS)은 하향식 발견 방법인 트리 형태의 병합을 위해 클러스터링 방법으로 구축하였다. 클러스터링 방법은 가장 긴 거리를 가지는 서열을 두 개의 클러스터로 나눌 수 있다는 것에 기초하였다. 제안한 새로운 서열 정렬 알고리즘은 기존의 Clustal W 알고리즘 보다 질적 향상과 처리 시간 단축 $O(n^3L^2)$이 기대된다.

ABSTRACT

Multiple sequence alignment(MSA) is a fundamental technique of DNA and protein sequence analysis. Biological sequences are aligned vertically in order to show the similarities and differences among them. In this paper, we propose an efficient group alignment method, which is based on clustering divergency, to perform the alignment between two groups of sequences. The proposed algorithm is a clustering divergence(CDMS)-based multiple sequence alignment and a top-down approach. The algorithm builds the tree topology for merging. It is also based on the concept that two sequences having the longest distance should be spilt into two clusters. We expect that our sequence alignment algorithm improves its quality and speeds up better than traditional algorithm Clustal-W.

▸ Keyword : multiple sequence alignment, clustering divergency

• 제1저자 : 이병일, 교신저자 : 이종연
• 접수일 : 2005.08.08, 심사완료일 : 2005.09.05
* 충북대학교 컴퓨터공학과, ** 충북대학교 컴퓨터교육과, *** 충북대학교 컴퓨터공학과
※ 이 논문은 2005년도 교육인적자원부 지방연구중심대학 육성 사업(제1세부과제)의 지원에 의해 연구되었음.

I. 서 론

다중 서열 정렬(multiple sequence alignment)은 수십년간 단백질과 핵산 서열들의 분석을 위한 중요한 도구로 발전되어 왔다[1,9]. 다중 서열 정렬은 패밀리(family) 분석, 계통관리(phylogeny) 분석, 도메인(domain) 분석 등의 기능 분석(functional analysis) 연구를 위해 매우 다양하게 사용된다[1,6].

일반적으로, 정렬에는 입력된 서열 전체를 비교하는 전역 정렬(global alignment)과 지역적으로 유사한 영역을 비교하는 지역 정렬(local alignment)로 나눌수 있다. 본 논문에서는 전역 정렬만을 고려한다.

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[4]. 즉, k개의 서열들을 정렬하는 일반적인 문제에서는 $O(n^k)$의 시간이 요구된다[5].

지금까지 다중 서열 정렬의 우수한 측정을 위해 많은 기준들이 제시되었는데, 그 중 가장 많이 사용되는 기준은 sum-of-pairs이다[6]. 이것은 Dayhoff나 Blosum 행렬[7]로 각각의 쌍 정렬의 점수를 산출하고, 쌍 정렬들의 모든 점수 합을 가지고 다중서열 정렬을 위한 점수를 생성한다.

따라서, 본 논문에서는 실용적인 방법으로 클러스터링 분기를 이용한 기반 다중 서열 정렬(CDMS) 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. 여기서, n은 서열들의 수, L은 모든 서열들 중에서 가장 긴 길이를 나타낸다. 실험 결과, Clustal W[8] 보다 정렬의 질과 실행 시간의 개선이 기대된다.

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구를 기술하고. 3장에서는 그룹 정렬 방법과 알고리즘의 동작 원리에 대하여 기술한다. 4장에서는 제안한 클러스터링 방법에 기초한 새로운 접근법을 소개하고, 5장에서는 실험 결과를 보여주었다. 마지막으로, 6장에서는 본 논문의 결론을 기술한다.

II. 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[7]. 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 포함
- 한 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_i와 같다.
  3. 오직 gap 문자만 가진 열은 존재하지 않는다.

가장 좋은 점수를 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.

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

여기서, c_1 은 일치, c_2 는 치환을 의미하며, c_3은 affine gap cost를 의미한다.

단백질 서열에서, 치환 행렬(TM)은 다름과 같이 사용한다.

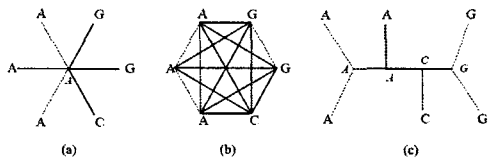
$$\delta(x,y)=\begin{cases} TM(x,y) \text{ if } x\neq gap \text{ and } y\neq gap, \\ 0 \quad\quad \text{if } x=gap \text{ and } y=gap, \\ affine\ gap\ cost \text{ if } x=gap \text{ or } y=gap. \end{cases}$$

여기서, TM은 Dayhoff나 Blosum 행렬[7]을 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[6]. 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]: k의 서열들 중 다중 서열 정렬 A에 대한 sum-of-pairs는 A의 모든 $\binom{k}{2}$ 쌍 정렬 점수 의 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[8]. 벌점 P_g는 갭의 시작과 관계된다. 또 다른 벌점 P_e는 갭의 길이와 관계된다. 따라서, 갭 벌점은 P_g+kP_e이다. 이것을 affined gap penalty 라고 부른다. 이 때 P_g와 P_e는 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. 두 개의 입력 서열 $a_1a_2...a_n$와 $b_1b_2...b_m$을 가정할 때,

1. $A(i,j)$는 $a_1a_2,...,a_n$와 $b_1b_2,...,b_m$의 최적 정렬 점수
2. $V(i,j)$는 $a_1a_2,...,a_n$와 $b_1b_2,...,b_m$의 마지막 a_i 와 b_j가 일치될 때의 최적 정렬 점수
3. $D(i,j)$는 $a_1a_2,...,a_n$와 $b_1b_2,...,b_m$의 마지막 a_i 와 공백이 일치될 때의 최적 정렬 점수
4. $I(i,j)$는 $a_1a_2,...,a_n$와 $b_1b_2,...,b_m$의 마지막 공백과 b_j가 일치될 때의 최적 정렬 점수

동적 프로그래밍 방법은 다음과 같은 초기값을 따른다.

$$\begin{aligned} &A(0,0)=0, \\ &A(i,0)=-P_g-iP_e,\ for\ i>0, \\ &A(0,j)=-P_g-jP_e,\ for\ j>0, \\ &D(i,0)=-\infty, for\ i>0, \\ &I(0,J)=-\infty, for\ j>0. \end{aligned}$$

그리고, $i > 0$ 와 $j > 0$일 때 계산은 다음과 같이 수행된다.

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

2.3 Clustal W를 이용한 다중 정렬

다중 서열 정렬을 위해 많이 사용하는 프로그램으로 Clustal W가 있다[8]. 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 정렬한다.

단백질 서열 정렬에 있어서 Clustal W의 휴리스틱 전략 가운데 하나는 각 정렬을 수행할 때 예상 진화 거리에 기반 한 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.

III. 경험적 방법을 이용한 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. 기본적인 방법은 하나의 그룹에 속한 각각의 서열에 동시에 gap을 같은 장소에 삽입한다. 그룹 정렬의 기본적인 알고리즘은 (그림 2)와 같다.

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

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, $1 \le i \le 8, 1 \le j \le 6$ 이다.

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

그림 2 그룹 정렬 알고리즘

서열 X_k, $1 \le k \le m$일때 $X_{k_1}X_{k_2}...X_{k_{L_1}}$로 표기하고, 서열 Y_l, $1 \le l \le n$ 일때 $Y_{l_1}Y_{l_2}...Y_{l_{L_2}}$로 표기한다. 여기에서 L_1와 L_2은 두 서열 X와 Y의 길이를 나타내며, $u()$은 PAM250와 같은 점수 행렬을 나타내고, α는 affine gap penalty를 나타낸다.

위 그룹 정렬에서 시간 복잡도는 $O(nmL_1L_2)$이다. 서열 S_k와 일련의 서열들 G사이의 거리는 $d(S_k, G)$로 표기한다. 이것은 S_k와 G의 모든 서열 사이에서 가장 짧은 거리를 정의한다.

위의 그룹 정렬 알고리즘을 가지고 그룹 X와 그룹 Y의 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 (그림 3)에서 보여준다. 각 과정의 계산식은 다음과 같다.

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

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

			-	-	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

그림 3 그룹 X(2) 와 그룹 Y(3)의 그룹 정렬 방법

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

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

(그림 3)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. 예를들어, $T_{8,6}$에서 $T_{7,5}$로 역추적할 때 정렬은 다음과 같다.

Group X
T
T
T
Group Y
T
-

$T_{6,4}$ 에서 $T_{5,3}$로 역 추적 정렬은 다음과 같다.

Group X
CTT
CTT
A-T
Group Y
ATT
A—

마지막으로, $T_{1,0}$에서 $T_{0,0}$로 역 추적 정렬은 다음과 같다.

Group X
AAGGCCTTT
-AGGGCTT
-AGGGA-T
Group Y
-TCG-A—
-AGGGCTT

최종 다중 서열 정렬은 다음과 같다.

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

Ⅳ. 제안하는 다중 서열 정렬 알고리즘

앞 장에서 그룹 정렬 방법에 대해 논하였다. 이 장에서는 트리 기반 방식에 기초한 CDMS 알고리즘을 제안한다. 제안한 CDMS 알고리즘(그림 4)의 기본적은 개념은 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 (그림 4)와 같다.

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

그림 4 CDMS 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_i와 S_j 서열을 서열 R에서 선택한다. 단계6에서는 R이 공백이 될 때까지 단계6.1~단계6.3을 실행한다. 단계7에서 입력이 S=G_1와 S=G_2이 될 때까지 CDMS 알고리즘을 반복하여 적용한다. 단계8에서 G_1와 G_2에서 그룹 정렬 방법을 수행한다.

L= max{| S_1|,| S_1|, . . . , | S_n|}라 할때, S_i의 길이를|S_i|라 표시하면 각 단계의 시간 복잡도는 다음과 같다.

- [단계 2]: $O(n^2L^2)$
- [단계 3]: $O(n^2 \log n)$
- [단계 6]: $O(n)$
- [단계 8] : $O(n^2L^2)$

거의 모든 경우 $L > n$ 이므로, 한번 반복할 때 $O(n^2L^2)$의 시간을 요구한다. 단계 7에서의 반복을 결합시키면 $O(n^3L^2)$의 시간 복잡도를 구할 수 있다.

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

S_1 = AAGGCCTT
S_2 = CGATT
S_3 = AGGGAT
S_4 = TCGA
S_5 = AGGGCTT

단계2 에서 얻은 거리 행렬을 〈표 1〉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 (그림 5)에서 보여 주고 있다.

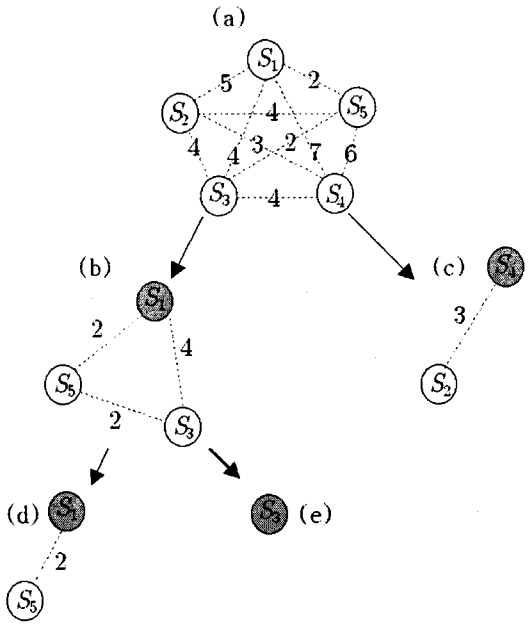


그림 5 클러스터링 방법

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 (c)는 (그림 5(a))로부터 분리된 것이다. 처음에, S_1와 S_4는 긴 거리 7을 가지기 때문에 $G_1=\{S_1\}$와 $G_2=\{S_4\}$이다. S_5는G_1에 가깝기 때문에, S_5를 G_1에 추가하면, $G_1=\{S_1, S_5\}$가 된다. S_3 또한 S_5와 가깝기 때문에 S_3도 G_1에 추가되어, $G_1=\{S_1, S_5, S_3\}$이 된다. 마지막으로, S_2은 S_4와 가깝기 때문에 S_2를 G_2에 추가시킨다. 마지막 클러스터링 결과는 $G_1=\{S_1, S_3, S_5\}$ 와 $G_2=\{S_2, S_4\}$ 이다.

그룹 G_1의 서열들의 수가 2보다 크기 때문에, G_1을 다시 분할하는 것을 (그림 5(d)) 와 (e)에서 보여주고 있다. (그림 5(d))와 (e)에서, S_1와 S_3가 가장 긴 거리 4를 가지고 있으므로, G'1 = $\{S_1\}$와 G'2 = $\{S_3\}$로 나눈다. S_5는 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.

V. 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(표 2). 다중 서열 정렬의 우수함을 결정하기 위해 sum-of-pair 측정을 사용하였고, 여기에 사용된 점수 행렬은 PAM250이다.

본 논문에서는 클러스터링 방법을 이용한 그룹 정렬 방법을 제안하였다. 제안한 알고리즘(CDMS)과 Clustal W의 성능 평가는 〈표 3〉와 같고, 두 알고리즘의 수행시간 비교는 (그림 6)과 같다. 실험 결과, 제시한 알고리즘이 Clustal W 보다 빠르고, 점수도 향상됨을 알 수 있다.

〈표 2〉 실제 생물학 테스트 데이터

	SOURCE
data1	Accession of NCBI: HBAQ, HACQ, B26543, HBGO, 23012, HBOR, P02067
data2	Accession of NCBI: NP_007567, CAC36948, NP_007373,NP_008100, NP_007386, NP_008217, NP_008230
data3	Accession of NCBI: AAK53588, NP_059474, NP_008289, NP_008659, NP_008342, NP_008225, NP_008212
data4	Accession of NCBI: NP_06622, AAG60030, NP_06621, NP_06620, NP_06542, NP_06396, NP_03825
data5	Accession of NCBI: AAK14328, AAK14327, AAG49582, NP_008649, NP_008279, AAK00949
data6	Accession of NCBI: NP_007571, NP_112728, NP_007377 NP_008104, NP_007390, NP_008234
data7	Accession of NCBI: NP_066226, NP_066204, NP_066204,NP_065431
data8	Accession of NCBI: NP_007568, NP_112528, NP_007374,NP_008108, NP_007387, BAA85281, NP_008231, NP_008049, AAG28222, BAA95622, AAK38695, NP_115356, NP_115434, NP_071648, NP_068788
data9	Accession of NCBI: AAK08547, AAK08571, NP_07164, NP_06878, NP_06655, NP_11213
data10	Accession of NCBI: NP_008342, NP_008225, NP_008212,NP_007381, NP_007368, NP_008095
data11	Accession of NCBI: NP_066558, BAB20723, NP_112096,NP_112109, NP_007565, NP_112122, NP_110507, BAB20736, AAF61381, BAB20710, NP_112525
data12	Accession of NCBI: NP_066226, NP_066206, NP_066204,NP_065431, NP_112525, NP_007565, NP_112122
data13	Accession of NCBI: NP_008345, NP_008215, NP_008228 BAA95619, AAK38692, AAB00992
data14	Accession of NCBI: AAK08554, AAK08578, NP_071645, NP_068785, NP_007371, NP_008098, NP_007384
data15	Accession of NCBI: NP_007565, NP_112122, NP_007371, NP_008098, AAB00992, AAK38692
data16	Accession of NCBI: NP_066223, AAG60030, NP_066216, NP_066201, NP_065428
data17	Accession of NCBI: NP_112522, NP_007562, NP_008342, NP_008225, NP_008212
data18	Accession of NCBI: NP_00824, NP_11433, AAK38689, CAC37083
data19	Accession of NCBI: NP_06396, NP_03825, NP_11437 AAK51683

〈표 3〉 CDMS와 Clustal W의 비교

	CDMS		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
data11	222826	1.187414	222826	1.451267
data12	68540	0.324586	68887	0.424164
data13	43004	0.259825	43004	0.399287

data14	59439	0.349632	594439	0.502054
data15	264897	1.344202	264897	1.627989
data16	40258	0.312156	40278	0.482699
data17	41375	0.342158	41375	0.555916
data18	26066	0.228225	26077	0.376905
data19	22973	0.200235	23012	0.333149
data20	24202	0.229099	24202	0.379132

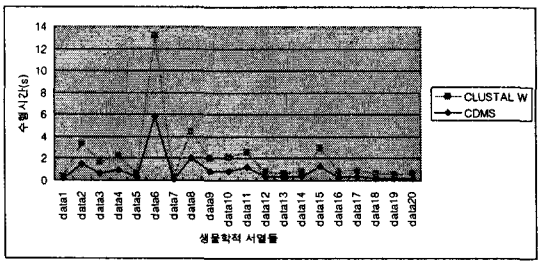


그림 6 CDMS와 Clustal W의 시간 비교

Ⅵ. 결 론

본 논문에서는 다중 서열 정렬 문제를 풀기 위해 CDMS 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 생각을 기반으로 하였으며, 최적의 MSA 문제를 해결하는데 $O(n^3L^2)$의 시간이 요구되는 CDMS 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] M, McNaughton, P, Lu, J, Schaeffer and D, Szafron, "Memory-Efficient A* Heuristics for Multiple Sequence Alignment, "Proceedings of Nineteenth National Conference on Artificial Intelligence (AAAI'2002),pages 737- 743, 2002.

[3] 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.

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

[5] 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.

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

[7] V. Bafna, E. L. Lawler, and P. Pevzner, "Approximation algorithm multiple sequence alignment," In In 5th Ann. Symp. on Pattern Combinatorial Matching, volume 807, pages 43-53, 1994.

[8] 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.

[9] 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.

저 자 소 개

이 병 일

1996년 2월 한발대학교 전자계산학과 공학학사

1998년 2월 충북대학교 컴퓨터공학과 공학석사

2001~2004 한국교원대학교 전산실 근무

2002~현재 충북대학교 컴퓨터공학과 박사 과정

〈관심분야〉 바이오인포메틱스, 다중 서열 정렬, 데이터마이닝

이 종 연

1985년 충북대학교 전자계산기공학과 (공학사).

1987년 충북대학교 대학원 전자계산기공학과 (공학석사).

1999년 충북대학교 대학원 전자계산학과 (이학박사).

1989년 비트컴퓨터(주) 개발부.

1990년~1994년 현대전자산업(주) 소프트웨어연구소 주임연구원.

1994년~1996년 현대정보기술(주) CIM사업부 책임연구원.

1999년~2003년 삼척대학교 정보통신공학과 조교수.

2003년~현재 충북대학교 컴퓨터교육과 부교수.

〈관심분야〉 질의 최적화, 시공간 데이터베이스, 데이터 마이닝, Bioinformatics, Ubiquitous Computing,GIS, u-learning.

정 순 기

1982년 8월 Uni. of Dortmund, Informatik, Dipl. Inform 취득

1994년 2월 Uni. of Groningen, Computing Science, Dr. 취득

1985년 5월~현재 충북대학교 컴퓨터공학과 교수

1994년 8월 충북대학교 전자계산소장

1998년 11월 한국과학재단 한독 기초과학협력위원회 정보분과 위원장

2000년 4월 충북대학교 도서관장

〈관심분야〉 데이터베이스 시스템, 소프트웨어공학, 소프트 실시간 시스템