DNA 마이크로어레이 데이타의 클러스터링 알고리즘 및 도구 개발

(Development of Clustering Algorithm and Tool for DNA Microarray Data)

여 상 수 † 김 성 권 ††
(Sang-Soo Yeo) (Sung-Kwon Kim)

요 약 DNA 마이크로어레이 실험으로 나오는 데이타는 아주 많은 양의 유전자 발현 정보를 담고 있기 때문에 적절한 분석 방법이 필요하다. 대표적인 분석 방법은 계층적 클러스터링(hierarchical clustering) 방법이다. 본 논문에서는 계층적 클러스터링의 결과로 나오게 되는 덴드로그램(dendrogram)에 대해서 후처리(post-processing)를 시행함으로써 DNA 마이크로어레이 데이타 분석을 더 용이하게 해주는 리프오더링(leaf-ordering)에 대해서 연구하였다. 먼저, 기존의 리프오더링 알고리즘들을 분석하였고, 리프오더링 알고리즘의 새로운 접근 방식을 제안하였다. 또한 이에 대한 성능을 실험하고 분석하기 위해서 계층적 클러스터링과 몇 가지 리프오더링 알고리즘들, 그리고 제안된 접근 방식을 직접 구현한 HCLO (Hierarchical Clustering & Leaf-Ordering Tool)에 대해서 소개하였다.

키워드 : DNA 마이크로어레이, 유전자 발현 정보, 계층적 클러스터링, 리프오더링

Abstract Since the result data from DNA microarray experiments contain a lot of gene expression information, adequate analysis methods are required. Hierarchical clustering is widely used for analysis of gene expression profiles. In this paper, we study leaf-ordering, which is a post-processing for the dendrograms output by hierarchical clusterings to improve the efficiency of DNA microarray data analysis. At first, we analyze existing leaf-ordering algorithms and then present new approaches for leaf-ordering. And we introduce a software HCLO(Hierarchical Clustering & Leaf-Ordering Tool) that is our implementation of hierarchical clustering, some of existing leaf-ordering algorithms and those presented in this paper.

Key words : DNA microarray, Gene Expression Profiles, Hierarchical Clustering, Leaf-Ordering

1. 서 론

생명공학의 발전은 짧은 시간 내에 대량의 생명체 정보들을 추출해 내는 데에 큰 기여를 하고 있다. 특히, 특정 환경 하에서의 많은 수의 유전자(gene)들의 발현도(expression level)를 분석해 내기 위한 고도의 생명공학 실험 기술인 DNA 마이크로어레이(microarray) 실험 기술은 대량의 유전자 발현 정보를 만들어 내게 된다. 이 DNA 마이크로어레이 기술은 유전자의 기능 예측(functional genomics), 유전자의 상호 관련성 분석(genetic networks)에 대한 연구에 많은 기여를 하리라 예상되고 있을 뿐만 아니라 돌연변이 검색, 질병의 진단, 신약 개발과 같은 보건 의료 분야로의 실질적인 기여도 가능할 것으로 예상되고 있다. 과학 기술적인 측면뿐만 아니라 인류의 건강과 생명의 신비를 해석하는데 아주 중요하다고 할 수 있다[1].

이러한 효과를 제대로 거두기 위해서는 DNA 마이크로어레이 실험을 통해서 나온 대량의 유전자 발현 정보들을 어떻게 더 이해하기 쉽고 분석하기 쉽게 가공하고 표현해 주느냐가 중요한 문제로 인식된다. 전산학적인 정보 가공과 효과적인 시각적 표현을 통하여 생물학자들이 쉽게 파악하고 연구할 수 있는 정보로 만들어 질 때, 비로소 대량의 유전자 발현 정보들이 의미가 있는 정보가 되는 것이다.

· 이 논문은 2001년도 한국학술진흥재단의 지원에 의하여 연구되었음 (KRF-2001-041-E00265)
† 비 회 원 : 중앙대학교 컴퓨터공학과
ssyeo@alg.cse.cau.ac.kr
†† 종신회원 : 중앙대학교 컴퓨터공학과 교수
skkim@cau.ac.kr
논문접수 : 2002년 10월 29일
심사완료 : 2003년 6월 24일

대량의 유전자 발현 정보들을 클러스터링(clustering)하는 문제는 이러한 배경에서 연구가 진행되어 왔다. 이 문제의 핵심은 발현 유형(expression pattern)이 서로 비슷한 유전자들은 서로 모으고 다른 발현 유형을 가지는 유전자들은 따로 분리해서, 결국은 몇 개의 클러스터(cluster)로 구별하자는 것이다. 클러스터링의 결과를 통해서, 일반적으로 알려진 기능의 유전자와 같은 클러스터에 속해 있는 밝혀지지 않은 유전자에 대해서 대략적으로 그 기능을 예측할 수 있다는 장점이 있다. 또한, 일반인과 환자에 대한 실험 결과를 서로 비교하면서 변화가 생긴 클러스터를 구분함으로써, 질병에 관련된 유전자 집단을 밝혀내는 데에도 도움이 된다.

유전자 발현 정보를 클러스터링하는 문제는 전산학 측면에서 풀기 어려운 문제에 속하기 때문에 여러 가지 휴리스틱 알고리즘들이 연구되어져 왔다. 현재까지의 연구에서는 그래프 이론이나 신경망 학습 기법, 통계학적인 다변량 분석 등을 사용해서 이 문제를 해결하고자 했다[1,2]. 그 중에서 현재 생물학자들이 가장 많이 사용하는 방법은 계층적 클러스터링(hierarchical clustering) 방법이다. 이 방법은 통계학에서 원래 연구된 방법으로서 Eisen에 의해서 유전자 발현 정보의 클러스터링 문제로 도입되었다[3]. Eisen은 계층적 클러스터링 기법을 이용한 소프트웨어 도구를 개발하였고, 이를 무료로 공개하여 생물학자들이 자신의 연구에 활용하도록 하고 있다[3,4].

계층적 클러스터링의 결과물은 흔히 덴드로그램(dendrogram)이라고 불리는 것으로서 노드들 간의 유사도(또는 거리)가 함께 표현된 이진 트리를 말한다. 최근에는 이 결과 트리의 리프노드(유전자를 의미함)의 순서를 전산학적으로 생물학적으로 더 의미 있게 재배열하는 방법에 대해서 여러 연구가 있었다[3-7]. 이 문제를 일반적으로 "리프오더링(leaf-ordering) 문제"라고 한다. 리프오더링 과정은 계층적 클러스터링의 결과에 대해서 전산학적인 의의를 더욱 부여하고, 클러스터링 결과의 직관성(intuition) 및 가시성(visibility)을 높이기 위해서 수행하는 후처리 과정(post-processing)이다. 일반적으로 계층적 클러스터링은 유사도가 가장 높은 한 쌍의 노드를 선택해서 합병해 가는 과정을 반복하게 되는데, 합병하는 과정에서 이미 합병된 노드들과의 새롭게 합병되는 노드들과의 관계는 고려하지 않는 단점이 존재한다. 리프오더링 과정을 거친 결과 트리는 연관성이 있는 유전자가 더 가까이에 위치하도록 되게 된다. 따라서 리프오더링 과정은 계층적 클러스터링의 확장된 개념에 포함되는 후반부 과정이라고 볼 수 있다.

1.1 리프오더링 문제의 정의

앞으로 논문에서 사용될 몇 가지 필요한 표기를 정의한다.

행렬 $D = \{d_{ij}, i=1,\ldots,m, j=1,\ldots,n\}$: DNA 마이크로어레이 실험 데이타로 구성된 $m \times n$ 행렬이다.

d_{ij} : i번째 유전자의 j번째 실험(어레이)에서의 발현도(expression level)를 의미한다.

d_i : i번째 유전자의 모든 실험(어레이)에 대한 발현도를 담고 있는 행벡터를 의미한다.

이진 트리 T : 계층적 클러스터링의 결과로 나온 이진 트리로써, m개의 리프노드를 가진다. 각 부모노드의 에지(edge) 길이는 양쪽 자식노드들 간의 유사도 정보를 나타낸다.

리프오더링 문제를 정의하면 다음과 같다.

덴드로그램으로 T를 그리면 루트노드를 맨 왼쪽에 위치시키고, 자식노드들을 부모노드의 오른쪽으로 그린다. 리프노드들은 맨 오른쪽에 수직으로 일렬로 그려진다. 이때 리프노드들의 순서를 리프오더링이라 부른다. T의 각 내부 노드의 두 자식노드 중에서 어느 것을 위쪽 자식노드로 하고 어느 것을 아래쪽 자식노드로 할 것인가를 덴드로그램을 그릴 때 정해야 한다. T가 $(m-1)$개의 내부 노드를 가지므로 모두 2^{m-1}개의 리프오더링이 가능하다. 이 중에서 목적 함수의 값을 최적으로 하는 리프오더링을 최적 리프오더링이라 부른다. 흔히 사용하는 목적 함수는 리프오더링에서 인접한 리프노드들 간의 거리(유사도의 반대 개념)의 총합을 최소화하는 것이다.

2. 기존의 리프오더링 알고리즘들과 그 접근 방식

2.1 기존의 리프오더링 알고리즘들

리프오더링 문제는 m의 크기(유전자의 수)가 매우 크기 때문에(대략 $10^2 \sim 10^3$개) 모든 경우의 수(2^{m-1}개)에 대한 전수 검색(exhaustive search)을 통해서 해결하는 것은 어려운 일이다. 따라서 여러 가지 휴리스틱 알고리즘들이 제안되었고[3-5], 최근에는 동적 프로그래밍을 이용한 최적해 리프오더링 알고리즘이 개발되었다[6,7]. 아래는 현재까지 연구된 리프오더링 알고리즘들을 설명한 것이다.

Eisen 알고리즘[3] : 첫 번째 방법은 휴리스틱한 방법의 대표적인 것으로서, DNA 마이크로어레이 데이타의 계층적 클러스터링을 가장 먼저 시도하였던, Eisen의 방법이다. 각각의 행벡터 d_i에 대한 모든 어레이들의 평균 발현도를 구해서, 노드를 합병할 때, 발현도가 낮은 노드를 위쪽에 두고, 높은 노드를 아래쪽에 두는 알고리즘이다.

Eisen-2 알고리즘[4] : 두 번째 방법도 역시 Eisen

이 제안한 휴리스틱 방법으로서, 그가 만든 소프트웨어인 Cluster의 사용자 설명서에서 제안한 것이다. 행렬 D에 대해서, 먼저 1차원 SOM(Self-Organizing Map)을 적용하여서, 클러스터링을 한다. 그리고 결과로 구분되어진 클러스터들에 대한 일련번호를 d_i에 기록한다. 그런 다음 다시 행렬 D에 대해서 계층적 클러스터링을 적용하되 두 노드를 합병하는 시점들마다 앞서 기록된 클러스터의 일련번호를 참조하여서, 최대한 SOM에서 만들어진 리프노드들의 순서가 유지되도록 하는 알고리즘이다.

Alon 알고리즘[5] : Alon이 제안한 휴리스틱한 방법이다. Alon은 확률적인 접근방식을 이용해서 하나의 클러스터를 이분화 해가는 반복적인 시도를 하는 새로운 계층적 분할 클러스터링 방법을 제안하면서, 또 다른 휴리스틱 리프오더링 방법을 고안하였다. 이 방법은 클러스터들을 분할해 갈 때, 새롭게 분할되어 가는 두 노드가 부모노드의 형제노드와 어느 쪽이 더 유사한 지를 확률적으로 계산해서, 유사한 쪽을 부모노드의 형제노드 쪽으로 배치하는 알고리즘이다. 그러나 이 방법은 Alon이 제안한 분할 방식의 계층적 클러스터링 방법에서만 적용될 수 있는 리프오더링 알고리즘이라는 제한성을 갖는다.

Yeo 알고리즘[8] : 본 연구자들이 제안했던 것으로서 각각의 행벡터 d_i 내에서 가장 발현도가 높은 실험의 번호를 d_i에 기록하고 두 노드를 합병하는 시점마다 기록된 번호를 참조하여 낮은 번호를 갖는 노드를 위로 위치하도록 하는 알고리즘이다.

랜덤 알고리즘 : 무작위로 위치를 결정하는 알고리즘이다.

최적해 알고리즘[6,7] : MIT의 Bar-Joseph 등과 Waterloo대학의 Biedl 등에 의해서 연구된 최적해 리프오더링 방법이다. 이것은 동적 프로그래밍 기법을 적용해서, 주어진 클러스터링 트리 T에 대해서 최적해를 구하는 리프오더링 방법이다. 처음 개발되었을 때에는 $O(n^4)$알고리즘이었지만[6], 그 다음 논문에서는 $O(n^3)$ 알고리즘을 연구 개발하였다[7]. 최적해 알고리즘은 전산학적인 면에서 가장 최적의 리프오더링을 찾아준다. 그러나 이것은 생물학적으로도 최적의 리프오더링이라는 것은 아니므로 위에 있는 휴리스틱 알고리즘들의 존재 가치는 여전히 남아 있다고 볼 수 있다.

2.2 기존의 리프오더링 알고리즘들의 접근 방식과 그 문제점

2.1절에서 언급된 리프오더링 알고리즘들은 알고리즘을 적용하는데 있어서, 공통적인 접근 방식을 가진다. 그것은 결과 트리 T의 전체 노드들에 대해서 한 번에 모두 리프오더링을 한다는 것이다. 이 접근 방식을 본 논문에서는 편의상 **전체적 리프오더링 접근 방식**이라고 부르기로 한다. 이것은 뒤에서 제안하게 되는 **클러스터별 리프오더링 접근 방식**과는 대조되는 개념이 된다.

전체적 리프오더링 접근 방식의 문제점을 설명하기 전에 먼저 계층적 클러스터링으로 나온 결과 트리 T를 가지고 어떻게 클러스터로 구분하는지를 먼저 설명하는 것이 필요하다. 계층적 클러스터링을 통해서 나온 결과 트리 T에 대해서, 일정한 값을 기준으로 에지를 자른다고 보자. 이때 자르는 기준이 되는 값을 절단치(cut-off value)라 부르기로 한다. 절단치는 유사도 값 범위 내에서 정하면 된다. 유사도는 피어슨 상관 계수(Pearson correlation coefficient)를 사용하고 있으므로 $[-1, 1]$의 범위를 가진다[1-3]. 절단치에 의해서 결과 트리 T는 결국 여러 개의 작은 트리들로 분리가 된다. 이렇게 분리된 작은 트리들이 바로 각각의 클러스터가 되는 것이다. 절단치가 -1에 가까울수록 클러스터의 개수는 적어지고, 절단치가 1에 가까울수록 클러스터의 개수는 많아진다.

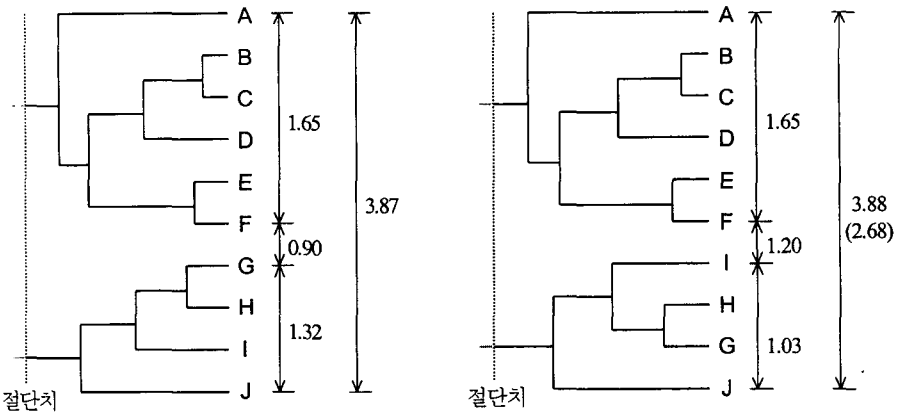
전체적 리프오더링 접근 방식은 이렇게 나누어지는 클러스터를 고려하지 않는다는 것이 문제이다. 자세히 설명하자면, 전체적 리프오더링 접근 방식은 클러스터 구분을 고려하지 않고 알고리즘이 진행된다. 따라서 자신의 클러스터가 아닌 다른 클러스터에 속해 있는 인접 노드들 사이의 거리도 고려해서 최소한으로 만들려고 하기 때문에 오히려 각 클러스터 내부의 정확한 리프오더링 결과를 보장할 수 없다는 것이다.

이런 문제점을 이해하기 쉽도록 한 가지 예를 들어서 설명해 보겠다. 그림 1의 (a)와 (b)는 처음 상태는 둘다 동일한 상태로서 계층적 클러스터링은 된 상태에서 특정한 절단치 값이 주어져서 유전자A~유전자F와 유전자G~유전자J가 각기 다른 클러스터에 속하되 인접한 클러스터인 상태였다고 가정한 것이다. 표 1은 이 예의 클러스터링과 리프오더링에 참조되는 유전자들(A~J) 사이의 유사도를 표로 만든 것이다. 표 2는 리프오더링의 결과를 설명하기 쉽게 표 1의 유사도를 거리로 환산한 것이다. 거리로 환산하기 위해서는 아래의 환산식이 사용된다.

유전자 간의 거리 = (1 - 유전자 간의 유사도)

표 1,2를 참조하여서 리프오더링을 진행해보자. 그림 1의 (a)는 전체적으로 최적해 리프오더링 알고리즘을 적용하여 나온 결과이고, (b)는 클러스터별로 최적해 리프오더링 알고리즘을 적용하여 나온 결과이다.

따라서 그림 1의 (a)의 유전자A~유전자J까지의 총 인접 거리의 합은 3.87이고 최적해이다. 반면에 그림 1의 (b)는 최적해는 아니다. 총 인접 거리의 합이 3.88이



(a) 전체적 리프오더링 접근 방식 (b) 클러스터별 리프오더링 접근 방식

그림 1 전체적 접근 방식과 클러스터별 접근 방식의 결과 차이의 예

표 1 그림 1에 대한 유사도 참조표

	A	B	C	D	E	F	G	H	I	J
A	1.00	0.40	0.20	0.10	0.06	0.04	-0.05	-0.07	-0.09	-0.09
B		1.00	0.90	0.65	0.50	0.30	0.07	0.03	0.00	-0.04
C			1.00	0.70	0.45	0.28	0.09	0.02	-0.02	0.00
D				1.00	0.50	0.32	0.10	-0.01	0.00	-0.10
E					1.00	0.85	-0.01	-0.04	0.01	-0.03
F						1.00	0.10	-0.10	-0.20	-0.21
G							1.00	0.80	0.64	0.57
H								1.00	0.60	0.41
I									1.00	0.28
J										1.00

표 2 그림 1에 대한 거리 참조표

	A	B	C	D	E	F	G	H	I	J
A	0.00	0.60	0.80	0.90	0.94	0.96	1.05	1.07	1.09	1.09
B		0.00	0.10	0.35	0.50	0.70	0.93	0.97	1.00	1.04
C			0.00	0.30	0.55	0.72	0.91	0.98	1.02	1.00
D				0.00	0.50	0.68	0.90	1.01	1.00	1.10
E					0.00	0.15	1.01	1.04	0.99	1.03
F						0.00	0.90	1.10	1.20	1.21
G							0.00	0.20	0.36	0.43
H								0.00	0.40	0.59
I									0.00	0.72
J										0.00

기 때문이다. 그러나 그림 1의 (b)에서는 첫 번째 클러스터(A,B,C,D,E,F)내에서의 결과가 1.65로써 최적해 상태를 보여주고 있고, 두 번째 클러스터(I,H,G,,J)내에서의 결과도 1.03으로써 최적해 상태를 보여준다. 이것은 클러스터 각각에 대해서 최적해 리프오더링 알고리즘을 적용했기 때문에 당연한 결과이다. 반면에 전체적 리프오더링 접근 방식으로 결과를 구한 그림 1의 (a)에서는 전체적으로는 최적해이지만, 두 번째의 클러스터(G,H,I,J) 내부는 최적해 상태가 아님을 알 수 있다. 이것은 근본적으로 두 클러스터 사이의 인접 부분에 위치하게 되는 유전자들 간의 거리는 고려할 필요가 없는 요소인데도 불구하고, 전체적 리프오더링 접근 방식에서는 이것까지도 포함해서 리프오더링을 시도하게 되므로 클러스터 내에서의 최적해 리프오더링을 구하지 못하는 문제를 초래하게 되는 것이다. 따라서 (b)와 같이 클러스터별로 리프오더링을 하는 것이 필요하며, 총 인접 거

리의 합에서도 두 클러스터 사이의 인접 부분에 위치하게 되는 유전자들 간의 거리는 제외시키는 것이 필요하다. (b)에서의 총 인접 거리 합은 3.88이 아니라 2.68로 보는 것이 바람직하다.

3. 클러스터별 리프오더링 접근 방식

2.2절에서 이미 언급한 바와 같이, 제안하는 리프오더링 접근 방식은 각 클러스터별로 리프오더링 알고리즘을 따로 적용하자는 것이다. 이 접근 방식을 본 논문에서는 편의상 클러스터별 리프오더링 방식이라고 부르기로 한다.

클러스터별 리프오더링 접근 방식은 간단히 다음과 같이 수행된다. 계층적 클러스터링의 결과 트리 T를 임의의 절단치를 정해서 클러스터별로 나눈다. 그 후에 각 클러스터별로 리프오더링 알고리즘들을 적용하면 된다. 전체 트리 T가 아니라 클러스터별 트리에 리프오더링 알고리즘을 적용하기만 하면 되기 때문에 이미 연구된 대부분의 리프오더링 알고리즘이 적용 가능하다. 그림 2는 이를 알기 쉽게 표현한 것이다.

제안하는 접근 방식을 사용하면, 앞서 말한 전체적 리프오더링 접근 방식에서의 단점으로 지적된 리프오더링이 필요 없는 구간을 리프오더링에서 제외시키게 되는 장점이 있다. 따라서 각 클러스터별로 알고리즘의 최선의 결과가 보장되는 것이다.

4. HCLO

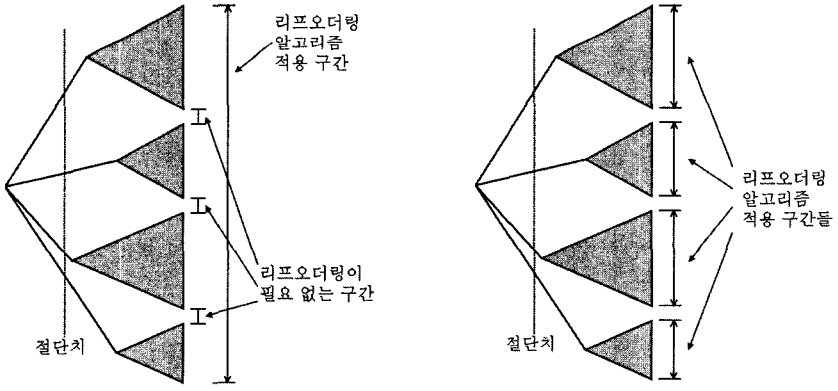
HCLO(Hierarchical Clustering and Leaf-Ordering Tool)는 본 논문에서 구현한 도구로서, 계층적 클러스터링 기능과 몇 가지 리프오더링 알고리즘들이 구현되어져 있다. 본 절에서는 HCLO에 대한 구체적인 설명을 하겠다.

4.1 기존의 도구들

HCLO이전에 연구 발표된 도구는 2가지가 있다. 첫 번째는 Eisen 등이 개발한 Cluster/TreeView 패키지[3,4]이고, 두 번째는 Biedl 등이 개발한 TreeArrange/TreePS 패키지[7]이다.

첫 번째 도구인 Cluster/TreeView 패키지는 클러스터링을 수행하는 Cluster와 클러스터링된 결과를 그래픽 화면으로 보여주는 TreeView로 구성되어있다. Cluster에는 계층적 클러스터링 방법, K-평균 방법, SOM (Self-Organizing Map) 등의 클러스터링 방법이 구현되어져 있다. Cluster의 결과는 2~3개의 결과 파일로 저장이 되고, 이 결과 파일을 TreeView에서 불러들여서 화면에 그래픽으로 표현해주게 되어있다. 물론 여기에는 리프오더링 알고리즘이 정식으로 포함되어 있지는 않다. 이 패키지는 MS Windows 응용프로그램으로 제작이 되었다. 개발 언어는 C/C++이며, 볼랜드사의 C/C++ Builder를 사용해서 컴파일하였다.

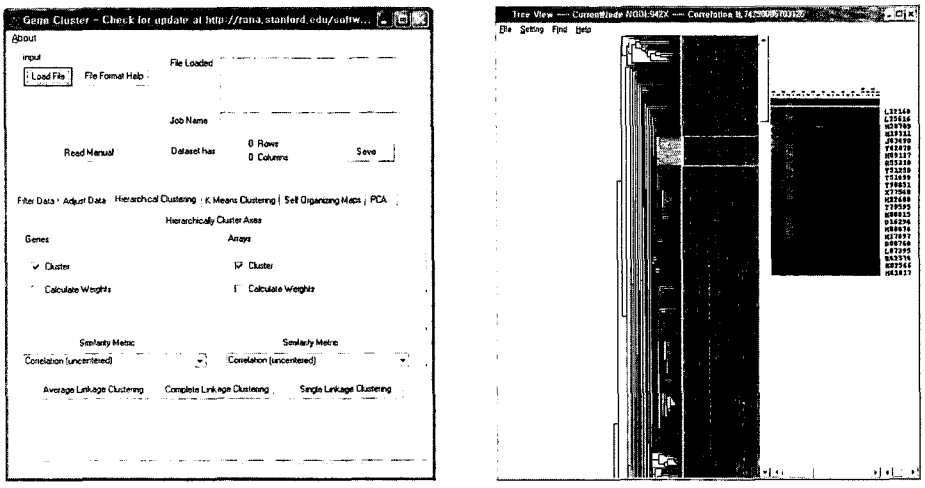
두 번째 도구인 TreeArrange/TreePS 패키지는 위에서 설명한 Eisen이 만든 Cluster의 출력 파일을 입력으로 해서 5가지의 리프오더링을 수행해 주는 TreeArrange와 TreeArrange의 결과를 포스트스크립트 (postscript) 파일로 만들어 주는 TreePS로 구성되어져 있다. TreeArrange는 단순히 입력 파일 내부의 유전자 순서만 바꾸어 주는 기능을 하기 때문에 TreeArrange의 결과를 TreeView에서도 100% 볼 수 있다. TreePS 또한 TreeView와 공통된 입력 형식을 가지기 때문에 TreeView로 볼 수 있는 것을 다 TreePS를 사용해서 포스트스크립트 파일로 만들어 주는 것이 가능하다. TreeArrange에는 클러스터링 기능이 포함되어 있지 않



(a) 전체적 리프오더링 접근 방식 (기존 방식)

(b) 클러스터별 접근방식 (제안하는 방식)

그림 2 기존의 접근 방식과 제안하는 접근 방식



(a) Cluster (b) TreeView

그림 3 Cluster와 TreeView의 실행 화면

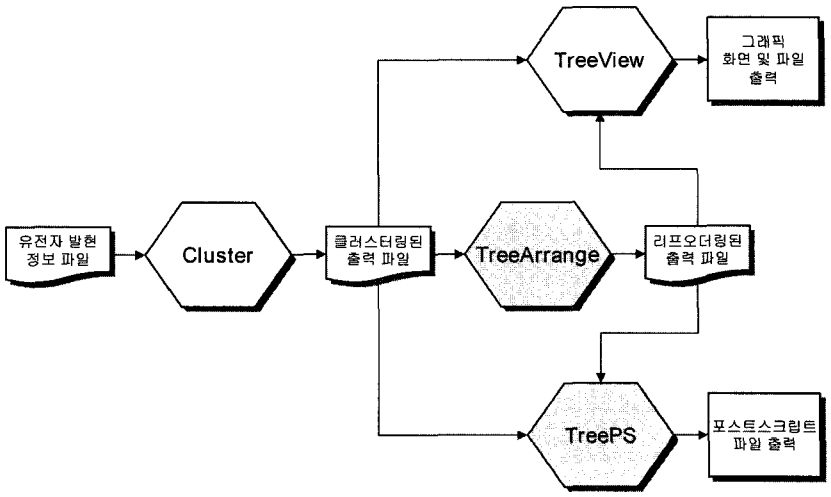


그림 4 Cluster/TreeView패키지와 TreeArrange/TreePS 패키지의 기능 및 관계도

으며, 전체적 리프오더링 접근 방식으로만 리프오더링 알고리즘들이 구현되어져 있다. TreeArrange에서 구현되어 있는 리프오더링 알고리즘은 랜덤 알고리즘, Eisen 알고리즘, 최적해 알고리즘이다. 이 패키지는 MS DOS/Windows 응용프로그램으로 제작되었고, 개발 언어는 C언어이다.

4.2 HCLO의 기능

HCLO는 앞서 말한 두 가지의 패키지와 마찬가지로 MS Windows 응용프로그램으로 제작되었고, 개발 언어는 C/C++언어이다. 컴파일러는 MS Visual C/C++를 사용하였고, 여기에 부가적으로 MFC(Microsoft Foundation Class Library)를 사용해서 사용자 인터페이스를 구현하였다.

HCLO는 Cluster의 계층적 클러스터링 기능이 완벽하게 구현되어 있다. 또한, 절단치를 주고 클러스터를 나눌 수 있는 기능이 추가되었다. 그리고 Eisen의 리프오더링 알고리즘, Random 리프오더링 알고리즘, Yeo의 리프오더링 알고리즘, 최적해 리프오더링 알고리즘이 구현되어 있다. 이 리프오더링 알고리즘은 앞 절에서 구분되어진 대로 전체적 리프오더링 접근 방식으로도 수행

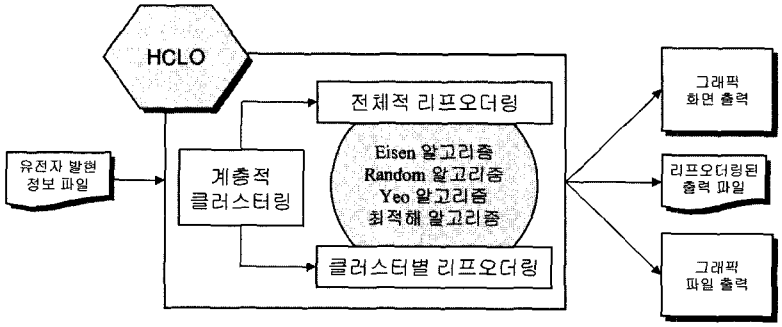


그림 5 HCLO의 기능 및 입출력

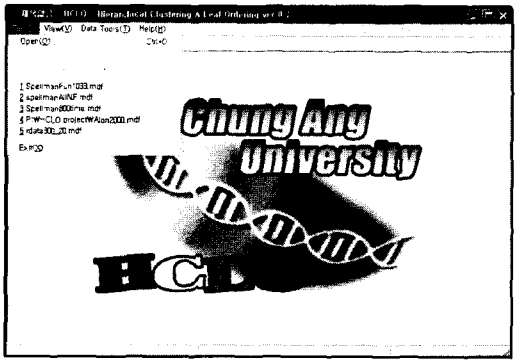


그림 6 처음 실행한 화면

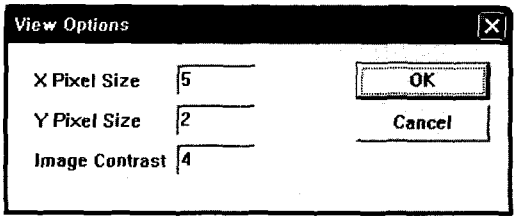


그림 7 유전자 발현 정보 표시 옵션 설정 대화 상자

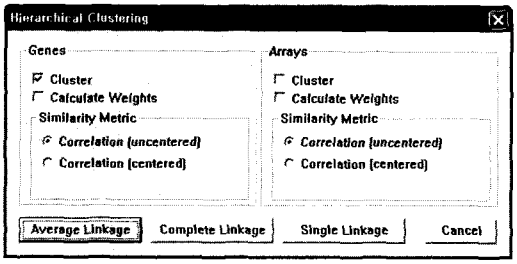


그림 8 계층적 클러스터링 설정 대화 상자

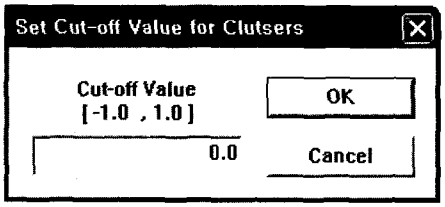


그림 9 클러스터를 나누기 위해서 절단치 설정 대화 상자

될 수 있고, 클러스터별 리프오더링 접근 방식으로도 수행될 수 있도록 설계되어 있다. 또한, TreeView나 TreePS와 같은 시각화를 위한 프로그램이 따로 필요하지 않고, HCLO 프로그램 자체에서 완벽한 그래픽 화면 출력이 지원된다. 클러스터별로 나누어진 상태도 명확하게 사용자가 확인할 수 있는 기능이 구현 되어져 있다. 그림 5는 HCLO의 기능을 간략하게 나타낸 것이고, 그림 6~그림 13까지는 HCLO의 실행화면을 갈무리한 것이다.

그림 6은 HCLO를 실행한 첫 화면을 보여준다. 파일 메뉴가 열려 있는 모습도 볼 수 있다. HCLO의 입력은 텍스트 형식의 유전자 발현 정보 파일이고, Cluster의 입력 양식과 동일하다.

그림 7은 하나의 유전자 발현 정보(즉 d_{ij}를 의미함)를 표현하는 옵션을 설정하는 대화 상자이다. 발현 정보는 점으로 표현되는데, 점하나의 x크기와 y크기를 정해주는 것이다. "Image Contrast"의 유전자 발현 정보를 표시하는 색의 농도를 정해주는 값이다. TreeView에서 설정해주는 것과 비슷하다.

그림 8은 계층적 클러스터링의 옵션을 설정해주고, 실행 시키는 대화 상자이다. 유전자에 대해서와 어레이에 대해서 클러스터링을 하는 것을 설정할 수 있다. 계층적 클러스터링 의 여러 가지 세부 방법인 average linkage, complete linkage, single linkage의 세 가지 모두를 제공하고 있다. 이것은 Cluster와 동일하다.

그림 9는 클러스터를 나누기 위한 절단치 설정 대화 상자이다. 여기서 절단치를 설정하면 클러스터링 결과 트리가 클러스터별로 분할된 형태로 보이게 된다. 이것은 HCLO에만 구현된 기능이다.

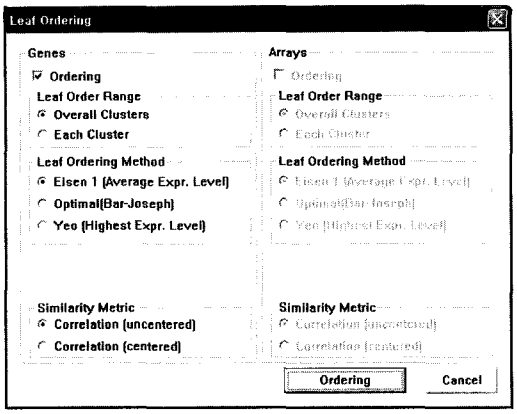


그림 10 리프오더링 설정 대화 상자

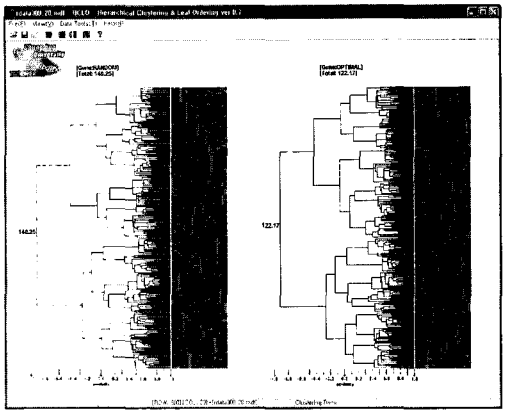


그림 12 전체적 리프오더링 접근 방식을 사용한 예

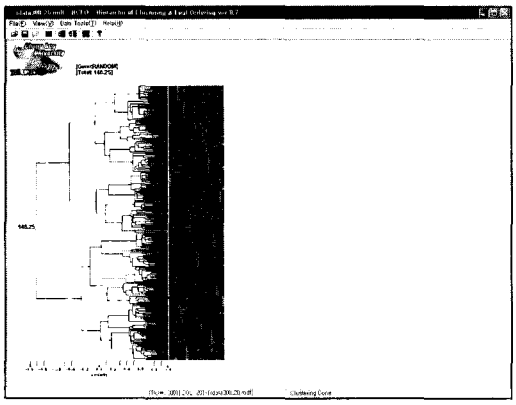


그림 11 계층적 클러스터링을 실행한 예

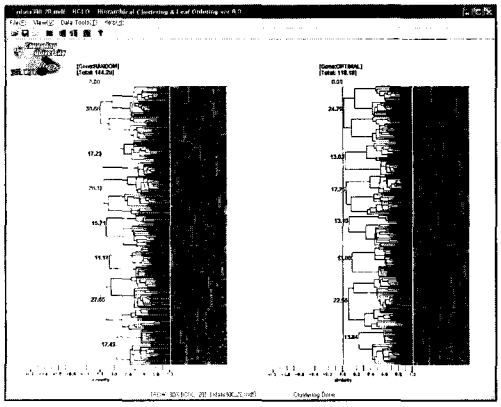


그림 13 클러스터별 리프오더링 접근 방식을 사용한 예

그림 10은 리프오더링의 옵션을 설정하고 실행하는 대화 상자이다. 위쪽의 라디오 버튼 그룹은 전체적 접근 방식과 클러스터별 접근 방식을 선택하는 부분이다. 이 역시 HCLO에만 구현된 기능이다. 가운데 쪽은 리프오더링 알고리즘을 선택하는 부분이다. 기본적으로 클러스터링을 할 때, 랜덤 알고리즘으로 리프오더링을 하기 때문에, 랜덤 알고리즘을 제외한 세 가지의 알고리즘을 선택할 수 있도록 되어 있다. 클러스터별 접근 방식으로 리프오더링을 하기 위해서는 먼저 절단치가 설정되어 있어야만 한다.

그림 11은 임의의 데이타에 대해서 계층적 클러스터링을 실행한 화면이다. 왼쪽의 결과 트리 아래에 나와 있는 안내자는 유사도(similarity value)의 정도를 나타내는 것이다.

그림 12는 그림 11의 결과에 대해서 전체적 접근 방식으로 최적해 알고리즘을 실행한 화면이다.

그림 13은 그림 11과 동일한 데이타에 대해서 클러스터별로 랜덤 알고리즘과 최적해 알고리즘을 실행한 화면이다. 이와 같은 결과는 그림 파일로도 저장이 가능하며, 리프오더링된 유전자를 텍스트 파일 형태로 열거하는 출력이 가능하다.

5. 실험 및 결과 분석

5.1 실험 데이타

실험 데이타는 표 3과 같은 총 7가지의 데이타를 사용하였다. 1),2),3)은 난수를 발생시켜 만든 후, Cluster를 이용해서 정규화와 로그 변환 등의 필터링을 거친

표 3 실험에 사용된 데이타들

	데이타 이름	유전자수	어레이수	비고
1	Random300_20	300	20	무작위 데이타1
2	Random500_20	500	20	무작위 데이타2
3	Random800_20	800	20	무작위 데이타3
4	Spellman800	800	82	효모 데이타1
5	Spellman523	523	82	효모 데이타2
6	Spellman1033	1033	82	효모 데이타3
7	Alon2000	2000	62	결장암 데이타

데이타들이다. 4)~6)은 Spellman[9]의 효모 데이타들 중에서 선택된 데이타들이다. 4)는 세포 주기(cell cycle)와 관련된 유전자 800개를 선택해서 만든 데이타이다. Bar-Joseph은 이 데이타들을 이용해서 생물학적인 분석을 하였다[6]. 5)는 cdc15에 대한 실험만을 선택한 데이타이다. 6)은 전체 효모 데이타 중에서 20%이상의 데이타가 없는(missing) 필드를 가지고 있는 유전자들과 적어도 로그 비율로 2를 넘는 발현도가 한 번도 나오지 않은 유전자들에 대해서는 필터링하고 난 뒤에 남은 523개의 유전자 데이타이다. 7)은 Alon[5]이 사용한 결장암(colon cancer)환자들에 대한 데이타이다.

5.2 실험 방법

다음의 방법으로 실험을 하였다.

5.2.1 전체적 접근 방식

• 사용한 리프오더링 알고리즘 : 랜덤 알고리즘, Eisen 알고리즘, Yeo 알고리즘, 최적해 알고리즘

5.2.2 클러스터별 접근 방식

• 나누는 클러스터의 개수 : 15~20개 사이가 되도록 절단치를 설정하였다. 생물학자가 실제 실험 데이타를 가지고 분석을 하는 경우에는 대부분 그 실험의 목적과 실험의 대상 등에 따라서 몇 개의 클러스터로 나누는지가 대략적으로 정해지게 된다. 그러나 본 실험에서는 전체적 리프오더링 접근 방식과 클러스터별 리프오더링 접근 방식의 성능 비교가 목표이기 때문에 이것은 잘 알 수 있도록 클러스터의 개수를 정하였다. 몇 번의 반복적인 실험을 통해서 클러스터의 개수가 많을수록 클러스터별 리프오더링 접근 방식에서 더 좋은 값이 나오는 것을 알 수 있었다. 그러나 클러스터의 개수를 많게 하기 위해서는 절단치를 낮게 설정해야 하고, 그렇게 되면, 하나의 리프노드만으로 이루어진 클러스터(Singleton)들이 많아지기 때문에 결과 비교가 적절하지 못한 경우가 발생했다. 이런 이유들을 바탕으로 15~20개 정도의 클러스터 개수가 두 접근 방식의 성능 비교를 위해 적절하다는 경험치를 얻었다.

• 전체적 접근 방식과 클러스터별 접근방식의 리프오더링 알고리즘 : 랜덤 알고리즘, Eisen 알고리즘, Yeo 알고리즘, 최적해 알고리즘

5.3 결과 분석

5.3.1 HCLO의 계층적 클러스터링 알고리즘과 리프오더링 알고리즘의 검증

HCLO를 사용해서 계층적 클러스터링을 한 후, 전체적 리프오더링 접근 방식으로 4가지 리프오더링 알고리즘을 실험한 결과가 표 4에 나와 있다. 여기에서 수치값은 인접 리프노드들 간의 거리를 모두 합한 것으로서, 작을수록 좋은 결과이다. 이 결과가 옳은지를 검증하기 위해서 Cluster와 TreeArrange를 이용했다. 이는 Cluster에는 계층적 클러스터링 방법이 구현되어 있고, TreeArrange에는 랜덤 리프오더링 알고리즘, Eisen 리프오더링 알고리즘, 최적해 리프오더링 알고리즘 등이 구현되어 있기 때문이다. 표 3의 동일한 데이타에 대해

표 4 전체적 리프오더링 접근 방식의 실험 결과

(각 항목의 수치는 결과 트리의 인접 리프 노드들 간의 거리 총합. 낮을수록 좋음)

데이타 \ 알고리즘	랜덤	Eisen	Yeo	최적해
Random300_20	149.18	147.65	147.61	122.17
Random500_20	226.56	227.11	224.27	184.73
Random800_20	344.59	343.60	343.75	281.19
Spellman800	331.59	333.71	331.26	289.31
Spellman523	235.51	232.99	232.48	199.82
Spellman1033	562.48	561.38	561.98	498.57
Alon2000	787.48	782.47	784.52	680.25

표 5 클러스터별 리프오더링 접근 방식의 실험 결과

(클러스터의 경계에 있는 리프노드들의 거리는 합산하지 않음)

데이타 \ 알고리즘	절단치	클러스터의 개수	랜덤	Eisen	Yeo	최적해
Random300_20	0.20	17	133.92	131.52	131.93	111.82
Random500_20	0.20	19	209.52	208.61	204.18	173.53
Random800_20	0.20	19	326.21	325.71	324.92	269.52
Spellman800	0.40	17	307.79	309.55	309.93	269.40
Spellman523	0.25	18	217.17	214.74	213.49	185.68
Spellman1033	0.30	16	538.68	537.21	537.01	478.62
Alon2000	0.35	19	764.55	758.69	758.41	660.45

서 Cluster를 사용해서 계층적 클러스터링을 수행한 후, TreeArrange를 통해서 리프오더링을 한 결과와 표 4의 결과를 비교해 보았을 때 일치하였다. 따라서 HCLO의 계층적 클러스터링 기능과 리프오더링 기능은 정확한 것으로 검증된다.

표 5는 HCLO를 사용해서 제안한 접근 방식인 클러스터별 리프오더링 접근 방식을 사용해서 실험한 결과이다. 여기서는 클러스터 경계에 위치하고 있는 리프노드들 사이의 거리는 합산하지 않았다. 이는 클러스터별 리프오더링에서는 클러스터 외부와의 거리 관계는 고려 대상이 아니기 때문이다.

전체적 리프오더링 접근 방식과 클러스터별 접근 방식의 결과 비교

이번에 살펴볼 내용은 실제로 전체적 리프오더링 접근 방식을 사용하였을 때와 클러스터별 리프오더링 접근 방식을 사용했을 때의 결과를 비교하는 것이다. 표 6은 최적해 알고리즘을 전체적 리프오더링 접근 방식으로 적용하고 절단치를 설정한 결과와 절단치를 준 다음 클러스터별 리프오더링 접근 방식으로 최적해 알고리즘을 적용한 결과에 대한 것이다. 결국 전체적 리프오더링 접근 방식과 클러스터별 접근 방식 모두 클러스터 내부의 거리만을 합산한 것이다. 원래는 전체적 접근 방식으로 분석하는 경우에는 절단치를 주고 클러스터 내부의 거리만을 구할 필요가 없지만, 두 접근 방식의 성능 비교를 위해 전체적 접근 방식으로 리프오더링을 한 후에 절단치를 설정한 후에 클러스터 내부의 거리 합산치를 구하였다.

표 6을 통해서 전체적 리프오더링 접근 방식으로 리프오더링을 했을 때보다 클러스터별로 리프오더링을 했을 때가 항상 합산치가 작은 것을 알 수 있다. 결국 클러스터별 리프오더링 접근 방식을 사용해서 최적해 리프오더링을 했을 때 각 클러스터별 최적해가 나온다는 것이 검증된다.

실제 생물 데이타에 대해서 어떤 결과가 나오는지를 자세히 보여주기 위해서 표 6의 실험 분석 결과들 중에서 Spellman800 데이타에 대해서 전체적 리프오더링 접근 방식과 클러스터별 리프오더링 접근 방식의 결과를 그림 14와 표 7로 나타내 보였다(Spellman800 데이타는 논문[9]에서 자세한 분석 설명을 위한 예로 사용되었던 대표적인 DNA 마이크로어레이 데이타이다). 그림 14는 두 접근 방식의 결과 화면을 갈무리한 것이고, 표 7은 이것을 표로 나타내어서 두 가지 결과에서 각 클러스터의 거리 합산치가 어떻게 변했는지를 대조한 것이다. 이것을 통해서 실제 생물 데이타에 대해서도 분명한 성능 향상이 있다는 것을 보여준다. 클러스터 중에서 하나의 리프노드만으로 구성되는 클러스터(singleton)는 클러스터별 리프오더링에 관련되지 않았으므로 비교대상에 넣지 않았다.

위의 두 가지 검증을 통해서 HCLO는 계층적 클러스

표 7 Spellman800 데이타에 대한 실험 결과 비교표

클러스터 번호	전체적 리프오더링 접근 방식(A)에서의 클러스터별 내부 거리 합산치	클러스터별 리프오더링 접근 방식(B)에서의 클러스터별 내부 거리 합산치	차이값 (A - B)
1	66.32	66.26	0.06
2	1.81	1.81	0
3	32.76	32.55	0.21
4	0.44	0.44	0
5	6.63	6.49	0.14
6	0.59	0.59	0
7	1.57	1.57	0
8	0.19	0.19	0
9	2.33	2.20	0.13
10	74.09	74.09	0
11	1.11	1.11	0
12	1.30	1.30	0
13	68.67	68.64	0.03
14	1.96	1.93	0.03
15	0.47	0.47	0
16	9.42	9.27	0.15
17	0.55	0.52	0.03
합계	270.16	269.40	0.76

표 6 최적해 알고리즘을 전체적으로 접근 방식으로 적용했을 때와 클러스터별 방식으로 적용했을 때의 각 클러스터별 인접노드 거리합의 합산치 비교

	최적해 리프오더링 알고리즘			
접근 방식 / 데이타	절단치	클러스터의 개수	전체적 리프오더링 접근 방식	클러스터별 리프오더링 접근 방식
Random300_20	0.20	17	112.65	111.82
Random500_20	0.20	19	174.52	173.53
Random800_20	0.20	19	271.29	269.52
Spellman800	0.40	17	270.16	269.40
Spellman523	0.25	18	186.49	185.68
Spellman1033	0.30	16	479.12	478.62
Alon2000	0.35	19	660.80	660.45

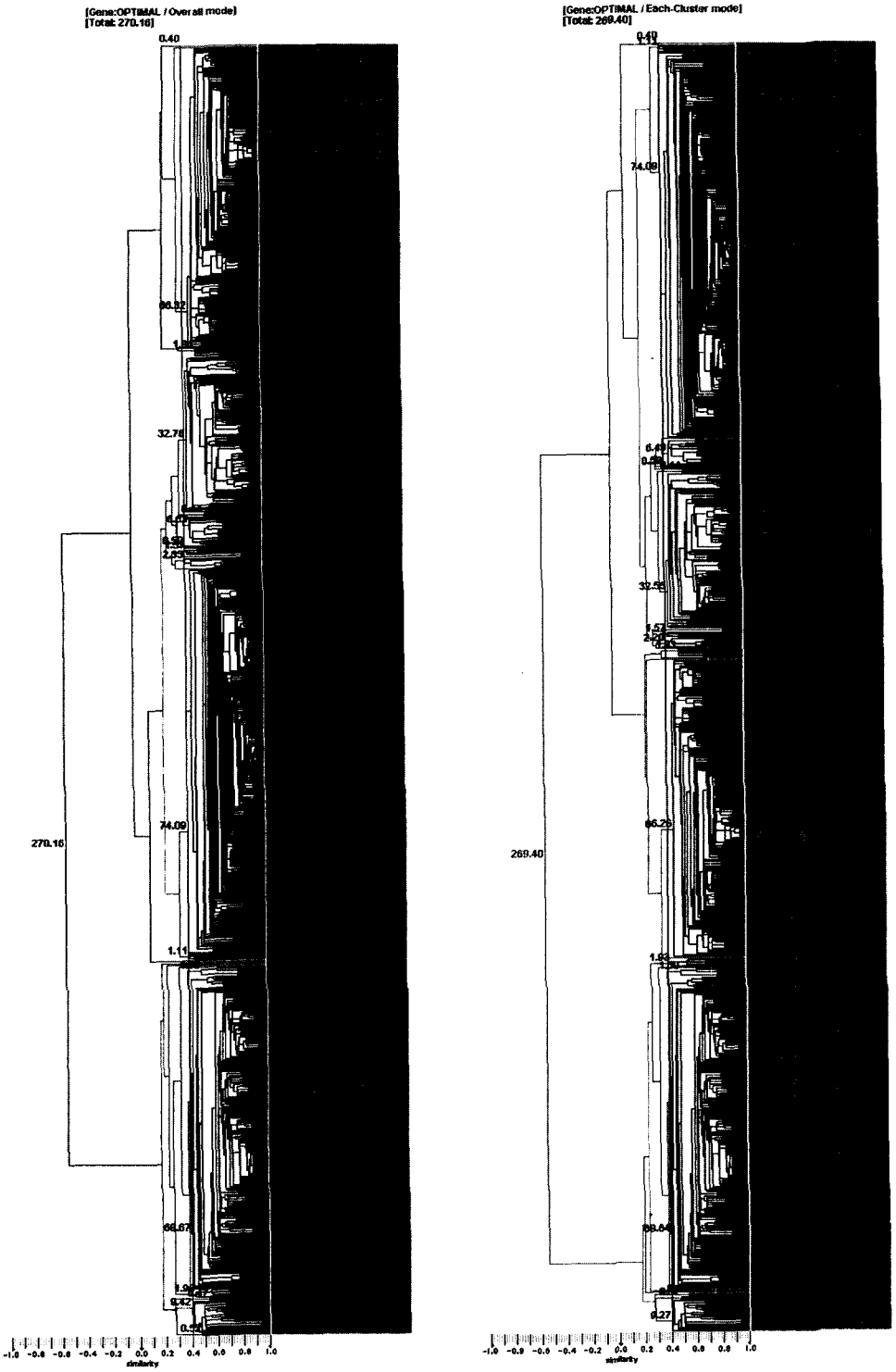


그림 14 Spellman800 데이타에 대해서 최적해 알고리즘을 전체적으로 접근 방식으로 적용했을 때(왼쪽)와 클러스터별 방식으로 적용했을 때(오른쪽)의 결과 화면

터링과 리프오더링 알고리즘을 정확하게 구현하였다는 것과 클러스터별 리프오더링 접근 방식이 전체적 리프오더링 접근 방식보다 적절하고 효율적이라는 것을 보여 주었다.

6. 결 론

본 논문에서는 현재까지 연구된 리프오더링 알고리즘들은 전체적 리프오더링 접근 방식을 사용한 알고리즘들이었다는 것과 그런 접근 방식의 단점이 무엇인지를 밝혀 보였다. 그리고 이러한 기존의 접근 방식의 단점을 보완하는 새로운 리프오더링 접근 방식인 클러스터별 접근 방식을 제안하였다. 대부분의 기존 리프오더링 알고리즘들은 클러스터별 리프오더링 알고리즘들로 변경해서 적용 가능하다.

제안한 클러스터별 리프오더링 접근 방식을 이용하는 리프오더링 알고리즘들을 실험하고 분석하기 위해서 새로운 소프트웨어 툴 *HCLO*을 구현하였다. *HCLO*는 Cluster의 계층적 클러스터링 기능과 TreeView의 그래픽 표현 기능을 통합한 분석 환경을 제공하며, 절단치를 설정해서 클러스터들의 상태를 보여주는 기능을 가지고 있다. 그리고 기존의 리프오더링 알고리즘들인 랜덤 알고리즘, Eisen 알고리즘, Yeo 알고리즘, 최적해 알고리즘들이 구현되어 있고, 이 알고리즘들을 클러스터별 접근 방식의 알고리즘으로 사용할 수 있도록 구현되어있다.

자체 구현한 HCLO를 사용하여 전체적 리프오더링 접근 방식으로 리프오더링 알고리즘들을 적용했을 때의 결과와 제안하는 클러스터별 리프오더링 접근 방식으로 리프오더링 알고리즘들을 적용했을 때의 결과를 비교 분석하여 제안하는 방식의 적절성과 효율성을 보여 주었다.

참 고 문 헌

[1] 여상수, 김성권, "DNA 마이크로어레이 데이타 클러스터링 알고리즘의 연구 동향", *한국정보과학회 컴퓨터 이론연구회지*, 제12권 1호, pp.2-11, 2001년 10월.

[2] R. Shamir and R. Sharan, "Algorithmic approaches to clustering gene expression data," *Current Topics in Computational Biology*, MIT Press, submitted.

[3] M. Eisen et al., "Cluster analysis and display of genome-wide expression patterns," *Proc. of Natl. Acad. Sci.*, 95:14863-14867, 1998.

[4] M. Eisen, "Cluster and TreeView Manual," *Eisen Lab. Homepage* (http://rana.lbl.gov/manuals/Cluster TreeView.pdf)

[5] U. Alon et al., "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Natl. Acad. Sci.*, 96:6745-6750, 1999.

[6] Z. Bar-Joseph et al., "Fast optimal leaf ordering for hierarchical clustering," *Proceedings of ISMB 2001*. pp.s22-s29.

[7] Z. Bar-Joseph, Therese Biedl, et al., "Optimal Arrangement of Leaves in the Tree Representing Hierarchical Clustering of Gene Expression Data," *Bioinformatics Research Group Homepage of University of Waterloo* (http://monod.uwaterloo.ca/supplements/01expr/art.pdf)

[8] 여상수, 이정원, 김성권, "DNA 마이크로어레이 데이타의 계층적 클러스터링에 대한 리프오더링 알고리즘 개발", *한국정보과학회 2002년 봄 학술발표논문집(A)*, 제29권 제1호, pp.706-708, 2002년 4월.

[9] P.T. Spellman et al. "Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization," *Molecular Biology of the Cell* 9:3273-97.1998.

여 상 수

1997년 2월 중앙대학교 컴퓨터공학과 학사. 1999년 2월 중앙대학교 컴퓨터공학과 석사. 2000년 3월~현재 중앙대학교 대학원 컴퓨터공학과 박사과정. 관심분야는 생물정보학, 컴퓨터이론, 암호 응용 및 정보보호

김 성 권

1981년 2월 서울대학교 계산통계학과 학사. 1983년 2월 한국과학기술원 전산학과 석사. 1990년 8월 University of Washington 전산학 박사. 1991년 3월~1996년 2월 경성대학교 전산통계학과 조교수. 1996년 3월~현재 중앙대학교 컴퓨터공학과 교수. 관심분야는 생물정보학, 계산기하학, 암호 응용 및 정보보호