유전체 상호간의 BLAST 최대 히트(best-hit)를 사용하여 서열화가 완성된 다수의 유전체로부터 Orthologous 단백질그룹을 자동적으로 클러스터링하는 기법

김 선 신[†] · 이 충 세[††] · 류 근 호[†††]

요 약

서열화가 완성된 유전체의 수가 최근에 빠르게 상승하고 있지만, 상동성에 의한 단백질 기능을 예측하는 방법은 충분히 연구되고 있지 않다. 서열화가 완성된 다수의 유전체로부터 유전체 상호간의 BLAST 최대 히트(best-hit)를 사용하여 OPCs(Orthologous Protein Clusters)를 만드는 일은 성공적으로 연구되어 왔다. 그러나 OPCs를 수작업으로 구축하는 것은 시간과 노력이 많이 드는 일이다. 이 논문에서 우리는 서열화가 완성된 다수의 유전체로부터 OPs(Orthologous Proteins)를 클러스터링하는 자동화 방법을 제시하고, 해당 클러스터링의 타당성을 수학적으로 증명한다.

키워드 : 서열화, 단백질 기능, 유전체, 상동성, 클러스터링

Automatic Orthologous-Protein-Clustering from Multiple Complete-Genomes by the Best Reciprocal BLAST Hits

Sunshin Kim[†] · Chung Sei Rhee[††] · Keun Ho Ryu[†††]

ABSTRACT

Though the number of completely sequenced genomes quickly grows in recent years, the methods to predict protein functions by homology from the genomes have not been used sufficiently. It has been a successful technique to construct an OPCs(Orthologous Protein Clusters) with the best reciprocal BLAST hits from multiple complete-genomes. But it takes time-consuming-processes to make the OPCs with manual work. We, here, propose an automatic method that clusters OPs(Orthologous Proteins) from multiple complete-genomes, which is, to be extended, based on INPARANOID which is an automatic program to detect OPs between two complete-genomes. We also prove all possible clustering mathematically.

Key Words : Genome, Protein Function, Orthologous Group, Automatic Method, Cluster

1. 서 론

최근 유전체 서열화의 빠른 성장으로 많은 양의 아미노산 서열이 단백질데이터베이스에 축적되고 있지만 그들 대부분의 기능이 밝혀지고 있지 않다. 특히 서열화가 완성된 유전체의 개수가 150이상으로 급격한 성장세가 지속되고 있음에도 불구하고 이들을 활용한 정보 추출기법들이 충분이 활용되지 못 하고 있다. 서열 유사도에 따라 유전자 또는 단백질서열의 상동성을 결정하고 단백질의 기능을 유추하는 것은 생물정보학의 기본 패러다임으로 여겨지고 있다. 생물학 실험실에서 기능이 알려지지 않은 단백질서열의 기능을 알고자 할 때, 이미 서열과 기능이 밝혀진 단백질로부터 새로운 서열의 기능을 유추하는 일은 매우 유용한 방법이 될 수 있다.

이와 같은 목적을 달성하기 위해 하는 방법 중의 하나가 OPCs(Orthologous Protein Clusters)를 만드는 일이다. 이 그룹은 다양한 기능을 하는 유전자 또는 단백질을, 동일한 기능을 하는 유전자 또는 단백질로 분류하여 저장함으로써, 이미 밝혀진 기능을 가진 단백질에 의해서 그렇지 않은 단백질의 기능을 유추 할 수 있다. 이렇게 각각의 기능별로 묶여진 군집들은 기능이 이미 알려진 단백질도 있고 그렇지

※ 이 논문은 2005년도 교육인적자원부 지방연구중심대학 육성사업의 지원에 의하여 연구되었음.
† 준 회 원 : 충북대학교 대학원 전자계산학과 박사과정
†† 정 회 원 : 충북대학교 컴퓨터과학과 교수
††† 종신회원 : 충북대학교 컴퓨터과학과 교수
논문접수 : 2005년 12월 19일, 심사완료 : 2006년 2월 2일

않은 것도 있다. 따라서 이미 알고 있는 단백질의 기능으로부터 아직 기능을 모르는 다른 단백질의 기능을 유추할 수 있다. 또는 새로이 생물학 실험으로부터 밝혀진 단백질(또는 유전자)서열의 기능을 추측하고자 할 때, 이미 구축된 OPCs와 서열을 비교하여 기능을 유추할 수 있다.

생물은 유전자 중복(duplication)과 종분화(speciation)를 통하여 진화한다고 가정할 수 있다. 이때 중복이 된 유전자는 종 분화 후 시간이 흐름에 따라 진화하여 서로 기능이 다른 것으로 변화 되는 것으로 여겨지고 있고, 다른 유전자들은 분화가 된 후 시간이 흐른 뒤에도 서로 같은 기능을 하는 것으로 믿어지고 있다. 시간이 지남에 따라 돌연변이(mutation)가 일어나면서 진화하여 유전자들의 서열 유사도는 점차 작아지게 된다. 또한 종 분화가 일어난 시기가 가까운 종들일 수록 서로간의 서열 유사도가 더 클 것이고 종 분화가 일어난 시기가 먼 종들일수록 서열 유사도가 작게 될 것이다. 이때 하나의 유전체 안에 존재하는 모든 유전자는 시간에 따라 진화하는 비율이 거의 같다고 가정할 수 있다. 따라서 종 분화가 일어난 이후 두 유전체간의 같은 기능을 하는 어떤 특정한 유전자의 서열 유사도는 다른 유전자들 보다 상대적으로 유사도가 클 것이라고 예상 할 수 있다. 이를 기반으로 현존하는 서열화가 완성된 유전체 사이에서 각각의 유전자를 상호 비교하여 최대 히트(best-hit)조건을 만족시키는 것들은, 서열 유사도가 가장 큰 것이므로, 같은 기능을 하는 유전자로 분류하여 하나의 orthologous 그룹으로 묶을 수 있다. 이렇게 종 분화에 의해서 진화된 유전자는 orthologs[1]라고 하여 기능이 같은 부류로 분류되어 왔고 중복된 후 종 분화된 유전자는 paralogs[1]라고 하여 기능이 다른 것으로 연구되어 왔다[2].

그런데 이와 같은 OPCs를 구축하는 과정에서 한 가지 장애물을 만나게 된다. OPs를 클러스터링하는데 있어서 생물학적 분석 및 수작업으로 인한 시간과 노력의 소모가 많다. 우리는 서열화가 완성된 여러 유전체로부터 자동적으로 OPs를 만드는 방법을 제시함으로써 이런 문제점을 극복한다.

2장에서는 관련연구를 살펴보고, 제안된 알고리즘은 3장에서 자세하게 다루어진다. 4장에서는 알고리즘의 성립에 필요한 모든 가능한 OPs를 수학적으로 증명하고 5장에서 결론을 내린다.

2. 관련 연구

Roman 등[3]은 박테리아, 아키아와 진핵생물(eukaryote)로부터 서열화가 완성된 21개의 유전체에 대해서 상호간에 최대 히트(best-hit)인 단백질은 기능이 동일 할 것이라는 Orthology 개념에 기반하여, BLAST[4] 프로그램을 사용하여 COGs(Clusters of Orthologous Groups of proteins)를 만들었다. 여기서 사용한 방법은 BLAST 프로그램을 사용하여 서열화가 완성된 유전체의 쌍으로부터 상호간의 최대 히트(best-hit)가 되는 단백질 쌍을 연결하여 삼각형 형태가 되는 그룹을 만든다. 그 다음으로는 삼각형에서 공통의 변을 가진 삼각형을 묶어주고 생물학적으로 분석하여 수작업으로 그룹을 분류하였다. 또한 임의의 임계값(threshold)을 사용하는 일 없이 클러스터링을 시행하였다.

NCBI의 COG 데이터베이스[3, 5]는 서열 지도가 완성된 66개의 유전체로부터 BLAST를 사용하여 각각의 쌍에서 최대 히트(best-hit)를 갖는 단백질을 동일한 기능을 하는 단백질로 가정하여 orthologous 그룹을 만들어 저장한 데이터베이스이다. BLAST 알고리즘은 휴리스틱(heuristic)방법을 사용하여 서열의 유사도를 측정하는 방법을 사용하고 있기 때문에 매우 효율적이고 빠른 장점이 있다[6-10]. 그러나 거리가 먼 종간의 유사도 측정에는 정확도가 많이 떨어지는 문제점을 안고 있다. 즉, 서열 유사도의 측정에서 점수(score)가 적은 경우에는 오류가 많이 일어나는 경향이 있다[11-14].

Kimmen등[14]은 계통발생학적(Phylogenetic)분석을 사용하여 단백질 기능을 예측하는 것이 더 적은 False positive를 산출한다는 것을 보여 주었다. 그러나 이 경우에는 여러 단계를 거치면서 더 많은 수작업(manual labor)이 필요하고 더 많은 컴퓨팅 파워를 요구한다.

KO(KEGG Orthology)[15]는 대사경로(metabolic pathways)와 조절경로(regulatory pathways)로부터 추출된 orthologous 유전자(또는 단백질)를 포함하여 orthologous 그룹을 테이블 형태로 보여 주는 데이터베이스이다. KO는 기능이 확실히 밝혀진 경로(pathways)로부터 수작업으로 작성되었기 때문에 기능별로 거의 정확히 분류되었다.

INPARANOID[16]는 BLAST를 사용하여 두 유전체 사이에서만 단백질을 비교하여 상호 최대 히트(best-hit)가 되는 단백질 a와 b를 찾아서 주된 Ortholog를 쌍으로 세우고, 이를 중심으로 in-paralogs(orthologs)를 추가적으로 클러스터링하는 일을 자동적으로 처리한 프로그램이다. 여기에 cut-off 값을 50bits로 하여 중요한지 않은 종간의 관계는 연결하지 않고 cut-off 중복(overlap)도 50% 이상 되는 것을 적용하여 짧은 도메인 수준의 일치(domain-level matches)가 되는 종사이의 연결을 이어주지 않았다. 여기에서 저자는 계통발생학적 방법(Phylogenetic method)과 자신의 결과를 비교하였다. INPARANOID는 계통발생학적 방법 보다 7%의 추가적인 orthologs를 발견하였으나 9%의 거짓-긍정(False-positive)과 3%의 거짓-부정(False-negative)은 증가함을 보여 주었다. 또한 참-긍정(True-positive)이 84%가 됨을 보여 주었다.

Montague등[17]은 COG에서 사용한 방식에 제한된 수(COG Stringency Number)를 추가적으로 도입하여 클러스터링 함으로써 유전자의 기능을 분류하여 각 유전자의 차이점을 나타내었다. 유전자의 기능들 사이에서 보존정도를 보여줌으로써 관련된 기능을 가진 유전자에 대한 실마리를 줄 수 있었다.

Stuart등[18]은 진화를 통해서 보존되는 유전자들의 기능이 전역적(Global)으로 관련되어 있다는 사실을 보이는데 활용하기 위해, 여러 유전체의 진핵생물로부터 메타진(metagenes)으로 정의한 Orthologous 그룹을 만들었다. 이들은 이

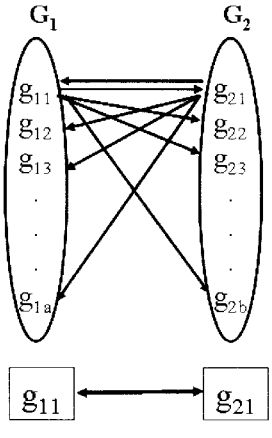
OPCs의 순도를 높이기 위해 E-value가 10^{-5} 이하가 되는 단백질만을 사용하여, COG에서 사용한 방법과 유사한 방법으로 클러스터링을 하였다.

이 논문에서 우리의 연구목표는 INPARANOID[16]에 기반 하여 서열화가 완성된 다수의 유전체로부터 자동적으로 OPCs를 구축할 수 있음을 수학적으로 보인다. 우리의 접근방식은 INPARANOID으로부터 나온 결과를 이용하여 모든 가능한 클러스터를 만든다. 따라서 INPARANOID 프로그램의 결과가 곧 우리의 OPCs의 질(quality)을 결정한다.

3. 제안한 클러스터링 방법

3.1 알고리즘의 기반

서열공간은 그래프로 표현 할 수 있다. 서열(sequence)은 노드(node)로 나타낼 수 있고 노드사이의 간선(link)은 두 서열 사이의 유사도로 나타낼 수 있다. 아래 (그림 1)에서 보여주듯이 유전체(Genome)들은 유전자(gene)들을 가지고 있으며, 각각의 유전체와 유전자를 다음과 같이 $G_1=\{g_{11}, g_{12}, g_{13}, ..., g_{1a}\}$와 $G_2=\{g_{21}, g_{22}, g_{23}, ..., g_{2b}\}$로 표현 할 수 있다. 한 쌍의 유전체간의 상호 최대 히트(best-hit)를 찾아서 연결하면 그들은 두 유전체간에 단백질 서열의 유사도가 가장 큰 것이 될 수 있다. 따라서 두 개의 단백질은 선형의 모양을 갖춘 같은 기능을 하는 그룹으로 분류하여 테이블 형태로 저장 할 수 있다.



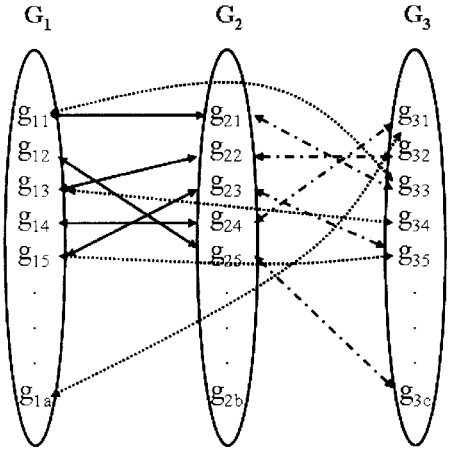
(그림 1) 상호 최대 히트(best-hit) 단백질 쌍(선형) 한개

정의 1. 만약 한 개의 단백질 쌍이 두 개의 유전체사이에 상호 BLAST 최대 히트(best-hit)를 가지면, 이 단백질 쌍은 한 개의 단백질선형을 이룬다.

정의 2. 만약 한 개의 클러스터가 공통된 단백질을 가진 두 개 또는 두 개 이상의 선형을 가지면, 이 클러스터는 한 개의 Orthologous 단백질 클러스터(Orthologous Protein Cluster)이다.

이제 간단한 예를 들어보자. (그림 2)에서 보여주는 것처럼 상호간의 단백질 쌍을 형성하는 세 개의 유전체가 있다고 하자. 두 개의 유전체 사이에 상호간의 BLAST 최대 히트(best-hit)를 이루는 단백질 쌍을 (그림 3)과 같이 저장 할 수 있다. 세 개의 테이블에서 공통의 단백질을 가진 단백질 쌍은 각각 동일한 OPC로 포함되어 (그림 4)와 같이 따로따로 저장 된다. 이때 한번 검색하여 저장된 단백질 쌍은 다시 반복하여 검색할 필요가 없도록 하기위해 플래그(flag)를 두어 표시해둔다.

이와 같이 일반적으로 n개의 유전체에 대해서도 각기 서로 상호간의 BLAST 최대 히트(best-hit)를 구할 수 있다. (그림 5)가 보여 주는 것과 같이, 이 경우 총 $n(n-1)/2$개의 모든 가능한 테이블을 생산하게 된다.



(그림 2) 세 개의 유전체 사이의 상호 최대 히트(best-hit) 단백질 쌍(선형)

T_{12}

G_1	G_2	Flag
g_{11}	g_{21}	yes
g_{12}	g_{25}	yes
g_{13}	g_{22}	yes
g_{14}	g_{24}	none
g_{15}	g_{23}	none

T_{13}

G_1	G_3	Flag
g_{11}	g_{33}	yes
g_{13}	g_{34}	yes
g_{15}	g_{35}	yes
g_{1a}	g_{31}	yes

T_{23}

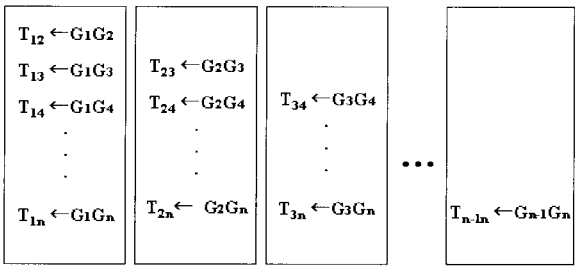
G_2	G_3	Flag
g_{21}	g_{33}	yes
g_{22}	g_{32}	yes
g_{23}	g_{35}	yes
g_{24}	g_{31}	yes
g_{25}	g_{3c}	yes

(그림 3) 세 개의 유전체 사이의 상호 최대 히트(best-hit) 단백질 쌍(선형)을 함유한 테이블

Γ_{123}

OPC001	g_{11}, g_{21}, g_{33}
OPC002	$g_{13}, g_{22}, g_{32}, g_{34}$
OPC003	g_{12}, g_{25}, g_{3c}
OPC004	g_{15}, g_{23}, g_{35}
OPC005	g_{1a}, g_{24}, g_{31}

(그림 4) 세 개의 유전체 사이의 상호 최대 히트(best-hit) 총괄 테이블



(그림 5) *n*개의 유전체 사이의 유전체 쌍들의 집합

3.2 클러스터링 알고리즘

제안하는 방법은, 단백질 서열의 유사도를 측정하는, BLAST 프로그램을 사용하여 서열화가 완성된 유전체간에 상호 최대 히트(best-hit)가 되는 단백질을 찾아 서로 연결하여 그룹을 만드는 일이다. 제안된 알고리즘은 첫 번째 단계로 n개의 유전체로부터 Orthologous 단백질을 클러스터링하는 것과, 두 번째 단계로는 이미 다 형성된 클러스터에 새로운 유전체를 추가하여 클러스터링하는 방법으로 나누어 생각할 수 있다.

1 단계 : 클러스터링하고자 하는 임의의 유전체 n개를 선택하여 모든 경우에 해당하는 상호간의 BLAST 최대 히트(best-hit)를 구한다. 이미 구한 선형 단백질 쌍으로부터 공통의 단백질을 가지고 있는 선형들을 하나의 같은 그룹으로 묶어준다.

2 단계 : 이와 같이 이미 형성된 OPCs에 새로운 유전체를 추가하기 위해서는 n개의 유전체와 새로운 유전체의 상호 BLAST 최대 히트(best-hit)를 사용하여 선형 단백질 쌍을 구한다. 그 다음 이미 형성된 OPCs와 선형 단백질 쌍을 비교하여 공통된 단백질을 가지고 있는 그룹에 포함시킨다.

(그림 6)는 임의의 선택된 n개의 유전체를 클러스터링하는 알고리즘으로 각각의 부 프로그램을 실행하는 절차를 보이고 있다. 특히 4번째 유전체부터 추가하여 n번째 유전체까지 클러스터링 할 수 있다. (그림 7)은 이미 완성된 n개의 유전체의 클러스터에 새로운 유전체를 추가하기 위한 알고리즘에 해당된다.

(그림 8)은 n개의 선택된 유전체에서 상호간의 유전체 사이에서 가능한 모든 쌍들의 상호 최대 히트(best-hit)를 구할 때 BLAST 프로그램을 사용하여 구하는 과정을 보이고 있다. (그림 9)는 새로운 유전체를 추가하는 경우 BLAST 프로그램을 사용하여 이전에 이미 사용한 n개의 유전체와 가능한 상호 최대 히트(best-hit)를 구하는 것을 보이고 있다. (그림 8)과 (그림 9)의 결과는 INPARANOID 프로그램으로부터 얻을 수 있다.

임의의 3개의 유전체에서 1번째 유전체와 2번째 유전체, 1번째 유전체와 3번째 유전체, 2번째 유전체와 3번째 유전체 사이에서 상호 최대 히트(best-hit)가 되는 단백질 쌍이 (그림 10)에서처럼 세 개의 테이블로 보여주고 있다. 이로부터 선형을 이루는 단백질 쌍에서 서로 공통인 단백질이 존재하는 경우를 찾아서 같은 그룹으로 분류하여 총괄 테이블 Γ에 저장하는 모든 가능한 과정을 (그림 15)가 보여주고 있다. 테이블 T에서 비교되는 공통의 유전체는 괄호 안에 아래첨자로 표시한다. 총괄 테이블 Γ에서 공통의 유전체는 아래첨자로 나타낸다. 교집합기호는 테이블사이의 공통된 단백질을 가진 쌍들을 총칭하기 위해 사용된다. 그리고 합집합기

When clustering *n* Genomes
Algorithm ClusteringGenomes

```
PairwiseSpeciesClustering
InitialOrthologClustering
for m=4 ... n
  OrthologClustering
end for
```

(그림 6) 유전체 클러스터링 알고리즘

When adding *(n+1)th* genome
Algorithm AddingNewGenome

```
PairwiseSpeciesAddClustering
m=n+1
OrthologClustering
```

(그림 7) 유전체 추가 클러스터링 알고리즘

Algorithm PairwiseSpeciesClustering
Require: *n* selected species

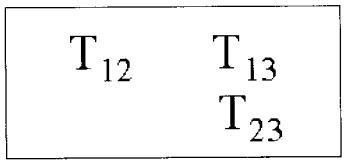
```
for i = 1 ... i = n-1 do
    for j=i+1 ... j=n do
        search the reciprocal best hits of two proteins in both Genome_i and Genome_j
        save the proteins in T_ij
    end for
end for
```

(그림 8) 유전체 사이의 상호 최대 히트(best-hit)를 구하는 알고리즘

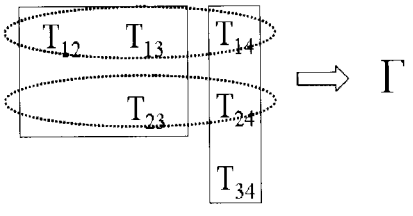
Algorithm PairwiseSpeciesAddClustering
Require: *(n+1) th* selected species

```
j = n+1
for i = 1 ... i = n do
        search the reciprocal best hits of two proteins in both Genome_i and Genome_j
        save the proteins in T_ij
end for
```

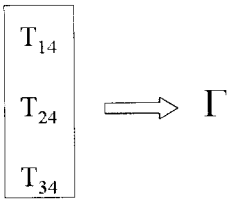
(그림 9) 추가하는 유전체의 상호 최대 히트(best-hit)를 구하는 알고리즘



(그림 10) 3개의 유전체로부터 형성된 테이블



(그림 12) 남겨진 테이블간의 평행적 관계



(그림 14) 남겨진 테이블간의 상호관계

Algorithm InitialOrthologClustering
Initial part
Require: Γ and T_{12}, T_{13}, and T_{23}

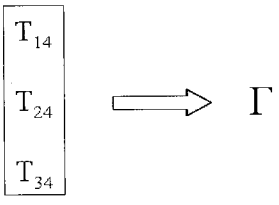
begin

$\Gamma_1 = \Gamma_1 \cup (T_{(1)2} \cap T_{(1)3})$
$\Gamma_2 = \Gamma_2 \cup (\Gamma_2 \cap T_{(2)3})$
$\Gamma_3 = \Gamma_3 \cup (\Gamma_3 \cap T_{2(3)})$
$\Gamma_2 = \Gamma_2 \cup (T_{1(2)} \cap T_{(2)3})$
$\Gamma_3 = \Gamma_3 \cup (\Gamma_3 \cap T_{1(3)})$
$\Gamma_3 = \Gamma_3 \cup (T_{1(3)} \cap T_{2(3)})$

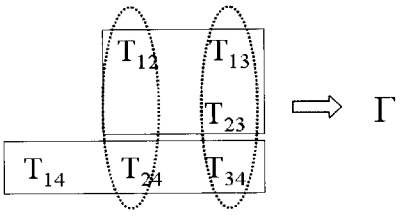
end

(그림 15) 임의의 3개의 유전체를 클러스터링하는 알고리즘

이제 추가되는 네 번째 유전체로 인해 생기는 새로운 테이블과 세 개의 유전체로부터 형성된 총괄테이블과 공통의 단백질을 가진 단백질 쌍을 저장하는 과정은 (그림 11)로 나타낼 수 있고, 다음 식으로 일반화 하여 표현된다. 이것은 알고리즘 (그림 16)에서 1행부터 7행까지의 수행절차와 같다. 모든 가능한 스텝은 식 (1)에서 case(a)와 case(b)로 두 가지 경우로 나누어 일반화 될 수 있다. Case(a)에서는 임의의 x개의 유전체로부터 형성된 총괄테이블과 x+1번째부터 추가되는 유전체로 인하여 생기는 테이블 T의, 즉 괄호안의 아래첨자가 나타내는 공통된 단백질을 가진 단백질 선형의, 결합관계를 나타낸다. Case(b)는 Case(a)에서 형성된 총괄테이블과 x+1번째로 추가되는 테이블 T의 괄호안의 m번째



(그림 11) 새로 생긴 테이블과의 관계



(그림 13) 남겨진 테이블간의 수직적 관계

유전체가 나타내는 공통된 단백질을 가진 단백질 선형의 결합관계를 나타낸다.

$$
\begin{aligned}
&Case\ (a): \Gamma_i^{n+1} = \Gamma_i^n \cup (\Gamma_i^n \cap T_{(i)m}) \\
&where \quad (i): 1 \le i \le m-1 \\
&Case\ (b): \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap T_{i(m)}) \\
&where \quad (m): 2 \le i \le m-1.
\end{aligned}
\tag{1}
$$

위와 같은 단계를 거치고 나면, (그림 12)와 같이 간단한 경우를 고려할 수 있다. 또한 다음 식의 두 가지 경우로 나누어 결합관계를 고려할 수 있다. Case(a)에서는 아직 단백질 쌍이 남아있는, x개의 유전체로부터 형성된, 테이블 T와, x+1번째부터 추가되는 유전체로 인하여 생기는, 테이블 T와의 평행적 관계를 나타낸다. Case(b)에서는 Case(a)에서 테이블 T들 사이에 공통인 단백질을 가진 단백질 선형의 결합체가 총괄테이블에 저장됨으로 인해서, Case(a)에서 사용되지 않은, x+1번째부터 추가되는 유전체로 인한, 나머지 테이블 T와 총괄테이블과의 공통인 단백질관계를 찾아서 결합하는 과정을 보여준다. 이 경우 (그림 16)에서 8행부터 20행까지는 다음과 같은 절차를 따른다. 이들 과정은 식 (2)로 일반화 된다.

$$
\begin{aligned}
&Case\ (a): \quad \Gamma_i^{n+1} = \Gamma_i^n \cup (T_{(i)j} \cap T_{(i)m}) \\
&where \quad (i): 1 \le i \le m-2, \quad i+1 \le j \le m-1. \\
&Case\ (b): \begin{cases} \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap T_{k(m)}) & when \quad (m): \ if \quad i \ne k \\ \Gamma_k^{n+1} = \Gamma_k^n \cup (\Gamma_k^n \cap T_{(k)m}) & when \quad (k): \ if \quad j = k \end{cases}
\end{aligned}
\tag{2}
$$

$$
where \quad 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad 1 \le k \le m-1.
$$

Algorithm OrthologClustering
Require: Γ and T_{ij}

```
1 begin
2   for i = 1 ... i = m-1 do
3       Γ_i = Γ_i ∪ (Γ_i ∩ T_(i)m)
4   end for
5   for i = 2 ... i = m-1 do
6       Γ_m = Γ_m ∪ (Γ_m ∩ T_i(m))
7   end for

8   for i = 1 ... i = m-2 do
9     for j = i+1 ... j = m-1 do
10        Γ_i = Γ_i ∪ (T_(i)j ∩ T_(i)m)
11      for k = 1 ... k = m-1 do
12        if i != k then
13          Γ_m = Γ_m ∪ (Γ_m ∩ T_k(m))
14          if j == k then
15            Γ_k = Γ_k ∪ (Γ_k ∩ T_(k)m)
16          end if
17        end if
18      end for
19    end for
20 end for
```

```
21 for i = 1 ... i = m-2 do
22   for j = i+1 ... j = m-1 do
23       Γ_j = Γ_j ∪ (T_i(j) ∩ T_(j)m)
24     for k = 1 ... k = m-1 do
25       if j != k then
26         Γ_m = Γ_m ∪ (Γ_m ∩ T_k(m))
27         if i == k then
28           Γ_k = Γ_k ∪ (Γ_k ∩ T_(k)m)
29         end if
30       end if
31     end for
32   end for
33 end for

34 for i = 1 ... i = m-2 do
35   for j=i+1 ... j=m-1 do
36       Γ_m = Γ_m ∪ (T_i(m) ∩ T_j(m))
37     for k = j+1 ... k = m-1 do
38       Γ_m = Γ_m ∪ (Γ_m ∩ T_k(m))
39     end for
40   end for
41 end for
42 end
```

(그림 16) 4번째 유전체부터 n번째 유전체까지의 OPs를 묶어주는 알고리즘

위 단계를 마치고 나면, 식(2)와 대조를 보이고 있는 다음 식을 고려할 수 있다. 이 경우에는 (그림 13)에서 보는바와 같이 남아있는 테이블 T들 간의 수직적 관계를 고려하여 Case(a)를 나타내고, 이로 인해 생기는 결합관계를 Case(b)에서 나타낸다. (그림 16)에서는 21행부터 33행까지 이와 같이 표현할 수 있다. 이 모든 과정은 식(3)로 일반화 될 수 있다.

$$Case\ (a): \Gamma_j^{n+1} = \Gamma_j^n \cup (T_{i(j)} \cap T_{(j)m})$$

$$where \quad (j): 1 \le i \le m-2, \quad i+1 \le j \le m-1.$$

$$Case\ (b): \begin{cases} \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap T_{k(m)}) & when \quad (m): \quad if \quad j \ne k \\ \Gamma_k^{n+1} = \Gamma_k^n \cup (\Gamma_k^n \cap T_{(k)m}) & when \quad (k): \quad if \quad i = k \end{cases} \tag{3}$$

$$where \quad 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad 1 \le k \le m-1.$$

마지막으로 처리할 수 있는 경우는 (그림 14)에서 보이는 것처럼, x+1번째부터 추가되는 유전체로 인한, 단지 남아있는 테이블 T사이에 결합관계를 Case(a)식으로 나타내고, 이로 인해 생기는 총괄테이블과의 결합관계를 Case(b)로 나타낸다. (그림 16)에서 34행부터 42행까지도 같은 절차를 따른다. 모든 가능한 스텝은 식(4)으로 일반화되어 표현된다.

$$Case\,(a): \Gamma_m^{n+1} = \Gamma_m^n \cup (T_{i(m)} \cap T_{j(m)})$$

$$where\,(m): 1 \le i \le m-2, \quad i+1 \le j \le m-1.$$

$$Case\,(b): \Gamma_m^{n+1} = \Gamma_m^n \cup (\Gamma_m^n \cap T_{k(m)}) \tag{4}$$

$$where\,(m): 1 \le i \le m-2, \quad i+1 \le j \le m-1, \quad and \quad j+1 \le k \le m-1.$$

4. 모든 가능한 클러스터링에 대한 증명

보조정리 4.1. *만약 n개의 서열화가 완성된 유전체로부터 OPs를 함유한 n(n-1)/2개의 테이블을 얻었다면, 이 테이블들의 전체 또는 부분합집합은 OPs으로 구성된다.*

증명. 이 보조정리는 자명하다. n개의 서열화가 완성된 유전체로부터 획득한 n(n-1)/2개의 테이블은 단지 OPs만을 포함하고 있으므로, 이 테이블의 전체 또는 부분 합도 단지 OPs만을 가지고 있어야 한다. □

보조정리 4.2. *만약 공통의 단백질을 가진 여러 개의 선형의 단백질쌍이 있다면, 한 개의 OPC(Orthologous Protein Cluster)는 이들 선형 단백질 쌍의 전체 또는 부분의 합으로 이루어진다.*

증명. 이 보조정리도 또한 자명하다. 만약 이들 선형 단백질쌍이 공통의 단백질을 가지고 있으면, 이들 단백질 쌍의 전체 또는 부분의 합은 하나의 OPC를 만든다. □

정리 4.1. *식(1)은 총괄테이블 Γ와 추가되는 유전체에 기인해서 생기는 새로운 테이블 사이의 모든 가능한 OPs를 묶어준다.*

증명. (그림 15)와 (그림 11) 그리고 다음 식을 고려하자.

$$\Gamma_1^{n+1} = \Gamma_1^n \cup (\Gamma_1^n \cap T_{(1)4})$$
$$\Gamma_2^{n+1} = \Gamma_2^n \cup (\Gamma_2^n \cap T_{(2)4})$$
$$\Gamma_3^{n+1} = \Gamma_3^n \cup (\Gamma_3^n \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$

알고리즘 (그림 15)는 세 개의 유전체로부터 얻어진 테이

블의 모든 가능한 결합을 나타낸다. 보조정리 4.1과 4.2에 의해서 위 식은 총괄테이블 Γ와 네 번째 유전체에 기인하여 생기는 새로운 테이블의 모든 가능한 결합을 나타낸다. 이 특정한 사실로부터, 위 식은 식(1)으로 쉽게 일반화 될 수 있다. □

정리 4.2. *식(2)은 이전 테이블과 추가적인 유전체에 기인하는 새로운 테이블 사이에 평행적 관계를 지닌 모든 가능한 OPs를 묶어준다. 그리고 총괄테이블 Γ와 추가적인 유전체에 기인하는 새로운 테이블의 나머지 사이에 모든 가능한 OPs를 묶어준다.*

증명. (그림 12)와 다음 식을 고려하자.

$$\Gamma_1^{n+1} = \Gamma_1^n \cup (T_{(1)2} \cap T_{(1)4}) \quad \Gamma_1^{n+1} = \Gamma_1^n \cup (T_{(1)3} \cap T_{(1)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)})$$
$$\Gamma_2^{n+1} = \Gamma_2^n \cup (\Gamma_2^n \cap T_{(2)4}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (\Gamma_3^n \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$

$$\Gamma_2^{n+1} = \Gamma_2^n \cup (T_{(2)3} \cap T_{(2)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)})$$
$$\Gamma_3^{n+1} = \Gamma_3^n \cup (\Gamma_3^n \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$

보조정리 4.1과 4.2, 위식에 의해, 세 개의 유전체로부터 생긴 여러 테이블과 네 번째 유전체에 기인한 테이블들 사이에 평행적 관계를 나타내고, 총괄테이블 Γ와 네 번째 유전체에 기인하여 생기는 새로운 테이블의 모든 가능한 결합을 나타낸다. 이들 식은 식(2)로 일반화된다. □

정리 4.3. *식(3)은 이전 테이블과 추가적인 유전체에 기인하는 새로운 테이블 사이에 수직적 관계를 지닌 모든 가능한 OPs를 묶어준다. 그리고 총괄테이블 Γ와 추가적인 유전체에 기인하는 새로운 테이블의 나머지 사이에 모든 가능한 OPs를 묶어준다.*

증명. (그림 13)과 다음 식을 고려하자.

$$\Gamma_2^{n+1} = \Gamma_2^n \cup (T_{1(2)} \cap T_{(2)4}) \quad \Gamma_3^{n+1} = \Gamma_3^n \cup (T_{1(3)} \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)})$$
$$\Gamma_1^{n+1} = \Gamma_1^n \cup (\Gamma_1^n \cap T_{(1)4}) \quad \Gamma_1^{n+1} = \Gamma_1^n \cup (\Gamma_1^n \cap T_{(1)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)}) \quad \Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)})$$

$$\Gamma_3^{n+1} = \Gamma_3^n \cup (T_{2(3)} \cap T_{(3)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{1(4)})$$
$$\Gamma_2^{n+1} = \Gamma_2^n \cup (\Gamma_2^n \cap T_{(2)4})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{2(4)})$$

보조정리 4.1과 4.2, 위식에 의해, 세 개의 유전체로부터 생긴 여러 테이블과 네 번째 유전체에 기인한 테이블들 사이에 수직적 관계를 나타내고, 총괄테이블 Γ와 네 번째 유전체에 기인하여 생기는 새로운 테이블의 모든 가능한 결합을 나타낸다. 이들 식은 식 (3)로 일반화된다. □

정리 4.4. *식(4)은 추가적인 유전체에 기인하는 새로운 테이블의 나머지들 사이에 모든 가능한 OPs를 묶어준다. 그리고 총괄 테이블 Γ와 추가적인 유전체에 기인하는 새로운 테이블의 나머지 사이에 가능한 모든 OPs를 묶어준다.*

증명. (그림 14)와 다음 식을 고려하자.

$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{1(4)} \cap T_{2(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (\Gamma_4^n \cap T_{3(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{1(4)} \cap T_{3(4)})$$
$$\Gamma_4^{n+1} = \Gamma_4^n \cup (T_{2(4)} \cap T_{3(4)})$$

이 식은, 보조정리 4.1과 4.2에 의해, 단지 네 번째 유전체에 기인하는 테이블 사이의 모든 가능한 결합을 나타낸다. 총괄테이블 Γ와 네 번째 유전체에 기인하여 생기는 새로운 테이블의 모든 가능한 결합을 나타낸다. 위 식은 식(4)로 일반화 된다. □

5. 결 론

서열화가 완성된 유전체로부터 동일한 기능을 하는 것으로 예측되는 단백질을 클러스터링하는 일은 새로운 단백질의 기능을 예측하는데 매우 유용하다. COG는 유전체간의 상호 BLAST 최대 히트(best-hit)를 사용하여 구축한 하나의 OPCs이다. 서열화가 완성된 유전체들의 수가 계속적으로 증가하는 시점에서, 이들을 생물학적 분석을 통하여 일일이 분류하는 것은 시간과 노력이 많이 드는 일이다. Remm 등은[16] 이 문제를 해결하기 위해 두 유전체 사이에 OPs를 찾아내는 자동화 프로그램을 만들었다. 우리는 이들의 일을 두개의 유전체로부터 다유전체로 확장하는 일을 했다. 서열화가 완성된 다수의 유전체로부터, 거의 수작업 없이, OPs를 묶어주는 자동화 방법을 제안했다. 즉, INPARANOID를 사용하여 서열화가 완성된 다수의 유전체로부터 Orthologous 단백질 선형을 획득하였다. 그리고 공통된 유전자를 가진 Orhologous 단백질 선형의 모든 가능한 결합을 수학적으로 증명하였다.

(그림 6)의 ClusteringGenomes 알고리즘은 $O(n^4)$의 시간 복잡도를 갖는다. n이 매우 클 때는 이런 방법이 적절하지 않다. 따라서 향후 연구로서 병렬 컴퓨팅이나 또는 다른 알고리즘을 써서 이 문제를 다룰 것이다.

참 고 문 헌

[1] W. M. Fitch, "Distinguishing homologous from analogous proteins", Syst. Zool., Vol.19, pp.99-113, 1970.

[2] RL Tatusov, et al., "A genomic perspective on protein families", Science, Vol.278(5338), pp.631-637, Oct., 24, 1997.

[3] L. Roman, et al., "The COG database : a tool for genome-scale analysis of protein functions and evolution", Nucleic Acids Research, Vol.28(1), pp.33-36, 2000.

[4] S.F. Altschul, et al., "Basic local alignment search tool", J. Mol. Biol., Vol.215, pp.403-410, 1990.
[5] RL Tatusov, et al., "The COG database: an updated version includes eukaryotes", BMC Bioinformatics, Vol.4(1), No.41, Sep., 11, 2003.
[6] S. A. Chervitz, et al., "Comparison of the complete protein set of worm and yeast:orthology and divergence", Science, Vol.282, pp.2022-2028, 1998.
[7] G. M. Rubin, et al., "Comparative genomics of the eukaryotes", Science, Vol.287, pp.2204-2215, 2000.
[8] S. J. Wheelan, et al., "Human and nematode orthologs-lessons from the analysis of 1800 human genes and the proteome of Caenorhabditis elegans", Gene, Vol.238, pp.163-170, 1999.
[9] A. R. Mushegian, et al., "Large-scale taxonomic profiling of eukaryotic model organisms : a comparison of orthologous proteins encoded by the human, fly, nematode, and yeast genomes", Genome Res., Vol.8, pp.590-598, 1998.
[10] M. Kanehisa & B. Peer, "Bioinformatics in the post-sequences era", nature genetics supplement, Vol.33, pp.305-310, 2003.
[11] P. Bork & E.V. Koonin, "Predicting functions from protein sequences-where are the bottlenecks?", Nat. Genet., Vol. 18, pp.313-318, 1998.
[12] J.A. Eisen, "Phylogenomics:improving functional predictions for uncharacterized genes by evolutionary analysis", Genome Res., Vol.8, pp.163-167, 1998.
[13] M. Y. Galperin & E.V. Koonin, "Sources of systematic error in functional annotation of genomes : domain rearrangement, nonorthologous gene displacement and operon disruption", In Silico Biol., Vol.1, pp.55-67, 1998.
[14] S. Kimmen, "Phylogenonmic inference of protein molecular function: advances and challenges", Bioinformatics, Vol.20, No.2, pp.170-179, 2004.
[15] H. Bono, et al., "Systematic Prediction of Orthologous Units of Genes in the Complete Genomes", Genome Inform Ser Workshop Genome Inform., Vol.9, pp.32-40, 1998.
[16] R. Maido et al., "Automatic Clustering of Orthologs and in-paralogs from Pairwise Species Comparisons", J. Mol. Biol., Vol.314, pp.1041-1052, 2001.
[17] G. Michael and A. H. Clyde, "Gene content phylogeny of herpesviruses", PNAS, Vol.98, No.10, May, 9, 2000.
[18] J. M. Stuart, et al., "A Gene-Coexpression Network for Global Discovery of Conserved Genetic Modules", Science, Vol.302, Oct., 10, 2003.

김 선 신

e-mail : sskim04@chungbuk.ac.kr
1989년 충북대학교 물리학과(학사)
1995년 충북대학교 대학원 물리학과 (이학석사)
2002년 시라큐스대 대학원 컴퓨터과학과 공학석사)
2003년~현재 충북대학교 대학원 전자계산학과 박사과정
관심분야 : 생명정보학, 데이터마이닝, 병렬컴퓨팅

이 충 세

e-mail : csrhee@chungbuk.ac.kr
1979년 University of South Carolina 대학원 컴퓨터과학과(이학석사)
1990년 University of South Carolina 대학원 컴퓨터과학과(이학박사)
1979년~1988년 University of North Dakota Assistant Professor
1989년~1991년 동아대학교 경영정보학과 부교수
1991년~현재 충북대학교 컴퓨터과학과 교수
관심분야 : 병렬컴퓨팅, 결함허용, 생명정보학, 정보보호

류 근 호

e-mail : khryu@dblab.chungbuk.ac.kr
1976년 숭실대학교 전산학과(이학사)
1980년 연세대학교 산업대학원 전산전공 (공학석사)
1988년 연세대학교 대학원 전산전공 (공학박사)
1976년~1986년 육군 군수 지원사 전산실(ROTC 장교), 한국전자통신연구소(연구원), 한국 방송대학교 전산학과(조교수) 근무
1989년~1991년 University of Arizona, Research Staff(TempIS 연구원, Temporal DB)
1986년~현재 충북대학교 컴퓨터과학과 교수
관심분야 : 시간 데이터베이스, 시공간 데이터베이스, 지식기반 정보검색 시스템, Temporal GIS, 데이터 마이닝 및 데이터베이스 보안, Bioinformatics 등