함수에 의한 정규화를 이용한 local alignment 알고리즘

(A Local Alignment Algorithm using Normalization by Functions)

이 선 호 † 박 근 수 ††
(Sunho Lee) (Kunsoo Park)

요 약 Local alignment 알고리즘은 두 문자열을 비교하여 크기가 l, 유사도 점수가 s인 부분 문자열 쌍을 찾는다. 크기가 충분히 크고 유사도 점수도 높은 부분 문자열 쌍을 찾기 위해 단위 길이당 유사도 점수 s/l을 최대화하는 정규화 방법이 제안되어있다. 본 논문에서는 증가함수 f, g를 도입하여 $f(s)/g(l)$을 최대화하는, 함수에 의한 정규화 방법을 제시한다. 여기서 함수 f, g는 DNA 서열을 비교하는 실험을 통해 정한다. 이러한 실험에서 함수에 의한 정규화 방법이 좋은 local alignment를 찾는다. 또한 유사도 점수의 기준으로 longest common subsequence를 채택한 경우, 기존의 정규화 알고리즘을 이용하면 별다른 시간 손실 없이 함수에 의해 정규화된 점수 $f(s)/g(l)$을 최대화 할 수 있음을 보인다.

키워드 : local alignment, 정규화, 함수에 의한 정규화, DNA 서열

Abstract A local alignment algorithm does comparing two strings and finding a substring pair with size l and similarity s. To find a pair with both sufficient size and high similarity, existing normalization approaches maximize the ratio of the similarity to the size. In this paper, we introduce normalization by functions that maximizes $f(s)/g(l)$, where f and g are non-decreasing functions. These functions, f and g, are determined by experiments comparing DNA sequences. In the experiments, our normalization by functions finds appropriate local alignments. For the previous algorithm, which evaluates the similarity by using the longest common subsequence, we show that the algorithm can also maximize the score normalized by functions, $f(s)/g(l)$ without loss of time.

Key words : local alignment, normalization, normalization by functions, DNA sequence

1. 서 론

Local alignment 알고리즘은 두 개의 문자열을 비교해 서로 유사한 부분 문자열 쌍을 찾아내는 알고리즘으로, 입력 받은 문자열 간의 모든 부분 문자열 쌍 (X, Y)중에서 유사도 점수 $S(X, Y)$가 가장 높은 (X, Y)쌍을 구한다. 유사도 점수 $S(X, Y)$를 정하는 기준으로는 X, Y사이에 공통으로 나타나는 부분 문자열 (LCS: Longest Common Subsequence) 길이나 X을 Y로 변환하는 문자의 삽입, 삭제, 치환과 같은 연산 개수 (edit distance) 등을 사용한다. 알고리즘은 또한 X, Y사이의 문자 단위 매핑도 구하는데, (X, Y)쌍과 그에 대한 매핑을 보통 local alignment라고 부른다. 이러한 local alignment는 두 가지 속성, 유사도 점수 $S(X, Y)$와 크기를 갖는데, 크기는 (X, Y)쌍의 문자열 길이의 합 $|X|+|Y|$로 정의한다.

Local alignment 알고리즘은 생물정보학 분야에서 DNA 문자열을 비교 분석하는 데 사용하고 있는데, 생물체의 DNA 문자열 중에서 서로 유사한 부분은 공통의 기능과 의미를 가질 것으로 추정하기 때문이다. 생물학적으로 의미 있는 local alignment를 찾기 위해서는 유사도 점수뿐만 아니라 크기도 동시에 고려해야 한다는 사실이 알려져 있다[1]. 이에 따라 단위 길이당 유사도 점수 $S(X, Y)/(|X|+|Y|)$와 같은 정규화된(normalized) 점수의 최대값을 구하는 알고리즘들이 제안되었다.

단위 길이당 유사도 점수를 최대화하는 기존의 방법은 Normalized Local alignment (NLA)[1]와 Normalized Longest Common Subsequence (NLCS)[2] 두 가지가 있다. 단위 길이당 유사도 점수 $S(X, Y)/$

· This study was supported by FPR05A2-341 of 21C Frontier Functional Proteomics Project from Korean Ministry of Science & Technology.
† 비 회 원 : 서울대학교 컴퓨터공학부
shlee@theory.snu.ac.kr
†† 종신회원 : 서울대학교 컴퓨터공학부 교수
kpark@theory.snu.ac.kr
논문접수 : 2005년 12월 7일
심사완료 : 2007년 3월 8일

($|X|+|Y|$)는 극단적인 경우 $X=a$, $Y=a$처럼 한 개 문자로 매치되는 경우 최대값을 갖지만 X, Y 길이가 짧아서 의미가 없다. 따라서 단위 길이당 유사도 점수도 높지만 충분한 크기를 갖는 local alignment를 구하기 위한 방법이 필요하다. NLA는 크기 $|X|+|Y|$가 너무 작은 alignment를 구하는 것을 막아주는 파라메터 L을 도입하여 변형된 유사도 점수 $S(X, Y)/(|X|+|Y|+L)$의 최대값을 구하며, NLCS는 유사도 점수 $S(X, Y)$가 기준값 T이상인 alignment 중에서 $S(X, Y)/(|X|+|Y|)$의 최대값을 구한다. 그렇지만 NLA의 경우 데이타에 따라 적절한 파라메터 L을 잡는 것이 어렵다는 것이 알려져 있고[1], NLCS의 경우 여전히 충분한 크기의 alignment를 구할 수 없다는 단점이 있다.

본 논문에서는 충분한 크기를 갖고 단위 길이당 유사도 점수도 높은 local alignment를 구하기 위해 함수에 의해 정규화된 local alignment(LANF: Local Alignment Normalized by Functions)를 제안한다. 증가함수 f, g를 이용해 정규화된 점수 $f(S(X, Y))/g(|X|+|Y|)$를 도입하면 인위적인 파라메터 L대신 함수 f, g가 정규화 정도를 정할 수 있다. 또한 적절한 함수 f, g를 이용하면 기존의 NLCS가 충분한 크기의 alignment를 구하지 못하는 단점을 해결할 수 있다. 유사도 점수 기준으로 LCS를 채택한 경우, 기존의 NLCS 알고리즘을 확장해 수행 시간에 손해를 보지 않고 LANF를 구하는 알고리즘을 얻을 수 있다.

논문의 구성은 다음과 같다. 2장은 기존의 정규화 방법을 분석하고 이를 토대로 3장에서 함수에 의한 정규화(LANF) 방법을 제안한다. 4장에서는 실제 DNA 서열 비교 실험을 통해 LANF 방법의 특징을 보인다. 마지막 5장에서 결론을 맺는다.

2. 기존의 정규화 방법

본 장에서는 기존의 정규화 방법 NLCS와 NLA를 비교 분석하여, 유사도 점수가 높고 매치되는 글자 쌍이 균일하게 분포하는 좋은 alignment를 찾는데 NLA가 더 적합한 방법임을 보인다.

정규화된 local alignment를 도입한 동기는 기존의 유사도 점수만을 고려하는 Smith-Waterman 알고리즘이 크기가 큰 alignment를 우선시한다는 문제점에서 출발한다[1]. 그에 따라 두 가지 문제점이 지적되었는데, 하나는 모자이크 효과(mosaic effect)로, 전체 점수는 높지만 내부에 유사도가 낮은 부분을 포함하는 alignment를 찾는 경우를 말한다. 다른 하나는 그림자 효과(shadow effect)로, 짧지만 유사도가 높은 alignment를 찾지 못하는 경우를 말한다. 비슷한 부분이 많은 DNA 서열을 비교하는 경우 이러한 문제점이 잘 발생하며 단위 길이당 유사도 점수와 같은 정규화된 점수를 도입하여 문제를 해결할 수 있다[1].

2.1 NLCS 정규화[2]

NLCS는 유사도 점수로 LCS 길이를 채택하고 $S(X, Y)/(|X|+|Y|)$의 최대값을 구한다. 너무 짧은 (X, Y)쌍을 구하지 않도록 $S(X, Y) > T$의 제약 조건, 즉 X, Y사이에 공통으로 나타난 문자 개수가 T개 이상인 (X, Y)쌍을 구하도록 하였다. NLCS는 충분한 크기의 alignment를 구하기 어려운 문제점이 알려져 있는데[2], 아래와 같이 간단하게 $S(X, Y) > 2T$인 alignment를 구할 수 없다는 것을 보일 수 있다.

Lemma 1 [2] 주어진 두 문자열에 대해, $S(X, Y) \geq T$ 이고 $S(X, Y)/(|X|+|Y|)$이 최대값을 갖는 부분 문자열 (X, Y)쌍은 항상 $S(X, Y) < 2T$를 만족한다.

증명. $S(X, Y)/(|X|+|Y|)$이 최대값을 갖는 (X, Y)쌍에 대해, $S(X, Y) \geq 2T$이면, (X, Y)를 쪼개서 더 좋은 (X', Y')을 얻을 수 있음을 보인다.

$X = X_1X_2$, $Y = Y_1Y_2$로 쪼개되, $S(X_1, Y_1) = T$, $S(X_2, Y_2) = S(X, Y) - T$ ㅁ T를 만족하도록 할 수 있다. $S(X_1, Y_1)/(|X_1|+|Y_1|)$, $S(X_2, Y_2)/(|X_2|+|Y_2|)$ 둘 중 큰 값을 갖는 쌍을 (X', Y')이라고 하자.

$$\frac{S(X',Y')}{|X'|+|Y'|} \geq \frac{S(X,Y)-S(X',Y')}{|X|+|Y|-(|X'|+|Y'|)}$$

분모는 0보다 크므로,

$$(|X|+|Y|)S(X',Y')-(|X'|+|Y'|)S(X',Y') \geq$$
$$(|X'|+|Y'|)S(X,Y)-(|X'|+|Y'|)S(X',Y')$$
$$(|X|+|Y|)S(X',Y') \geq (|X'|+|Y'|)S(X,Y)$$
$$\frac{S(X',Y')}{|X'|+|Y'|} \geq \frac{S(X,Y)}{|X|+|Y|}$$

(X', Y')의 단위 길이당 유사도 점수는 항상 (X, Y)보다 크거나 같다. □

비교하려는 두 서열 사이에 $2T$ 이상의 유사도 점수를 갖고 매치되는 글자가 고르게 분포하는 alignment가 존재한다고 하자. Lemma 1에 따르면, NLCS 알고리즘은 이러한 좋은 alignment를 쪼개어 그 일부분만 답으로 구하게 된다.

2.2 NLA 정규화[1]

NLA 정규화는 유사도 점수로 Smith-Waterman 알고리즘의 유사도 점수를 채택하였고, 정규화 정도를 정해주는 파라메터 L을 도입해 $S(X, Y)/(|X|+|Y|+L)$의 최대값을 구한다. 파라메터 L이 클수록 구하는 (X, Y)쌍의 크기는 커지고, L이 작을수록 (X, Y)쌍의 크기도 작아지는 경향이 있다.

NLA 정규화에서는 alignment내에 매치되는 글자가 고르게 분포하는 경우, 해당 alignment를 쪼개서 얻은 sub-alignment가 더 높은 점수를 받기 어렵기 때문에

Lemma 1과 같은 문제점은 발생하지 않는다. Alignment의 유사도 점수 $S(X, Y)$를 s, 크기 $|X|+|Y|$를 l이라고 했을 때, 단위 길이당 유사도 점수 s/l을 k라 하자. 그러면 NLA 정규화 점수는,

$$\frac{s}{l+L}=\frac{kl}{l+L}=\frac{k}{1+\frac{L}{l}}$$

꼴이 된다. 이를 크기 l에 대한 함수로 본다면 그림 1-a와 같이 위로 볼록한 증가 함수가 된다. 이 함수는 l이 증가함에 따라 증가율이 감소한다. 그림 1-a와 같이 l이 커질수록 함수값이 k에 의해 쉽게 증가하고, l이 작을수록 함수값은 k보다 l에 의해 쉽게 증가한다. 따라서 NLA 점수는 l이 작은 구간에서 k의 영향이 적고, 단순한 l에 대한 증가 함수에 가까워진다.

NLA 정규화를 매치되는 글자가 고르게 분포하는 alignment에 적용한다면, 전체 alignment의 k값과 sub-alignment들의 k값은 비슷하다고 가정할 수 있다. 이 경우 l이 작은 sub-alignment일수록 크기가 큰 전체 alignment에 비해 높은 NLA 점수를 받기 어렵다.

또한 NLA 정규화에서 파라메터 L은 NLA 점수의 증가율을 조정하여 정규화 정도를 결정한다. 그림 1(b)와 같이 L이 작을수록 NLA 점수는 빨리 증가하여, k보다 l에 의해 높은 점수를 받기 쉬운 구간은 작아진다.

3. 새로운 함수에 의한 정규화(LANF) 방법

본 장에서는 NLA 정규화 점수가 alignment 크기 l에 대한 위로 볼록한 증가함수라는 분석 결과를 토대로 LANF 방법을 제안한다. 또한 기존의 NLCS 알고리즘을 확장하면 LANF 알고리즘을 얻을 수 있음을 보인다.

3.1 LANF 정규화

LANF 정규화 방법은 NLA와 마찬가지로, 단위 길이당 유사도 점수 k가 높다면 alignment 크기 l이 클수록 이득을 보는 효과를 내면서, 그 효과를 외부에서 정해주는 파라메터 L대신 미리 정의된 함수에 의해 결정하는 것이 목적이다. 이를 위해 증가함수 f, g를 이용해 LANF 점수 $f(s)/g(l)$를 정의한다. NLA와 마찬가지로 s/l이 k로 같은 경우 $f(kl)/g(l)$은 l에 대한 증가함수이며 위로 볼록한 함수가 되도록 한다. 이 경우 $f(s)$는 $g(l)$보다 빨리 증가하는 함수여야 한다. 이러한 함수의 예로는 $s\log s/l$, $s\sqrt{s}/l$, $s^{\frac{n+1}{n}}/l$ 등이 있다.

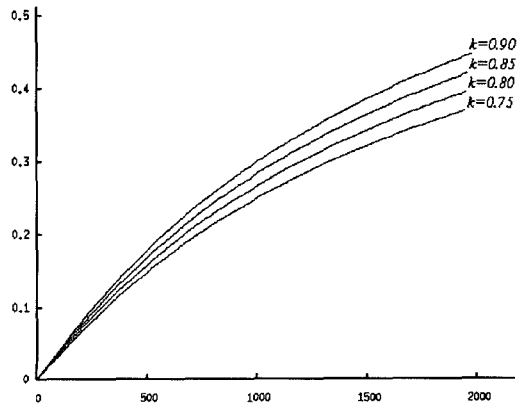
좋은 alignment를 찾기 위한 LANF 정규화에 적합한 함수 f, g는 실험을 통해 결정한다. 좋은 alignment의 단위 길이당 유사도 점수 s/l와 크기 l사이의 관계는, 본 논문에서 분석한 NLA 점수가 위로 볼록한 증가함수라는 결과 외에 알려진 것이 없다. 따라서 LANF 정규화 점수는 NLA와 비슷한 결과를 보이는 함수를 선택한다. LANF 정규화 점수를 찾기 위해 $s\log\log s/l$, $s\log s/l$, $s^{\frac{8}{7}}/l$ 등의 다양한 위로 볼록한 증가함수를 시도해 보았고, 그 중 가장 NLA와 비슷한 결과를 보이는 것이 $s\log\log s/l$이다.

3.2 NLCS 및 LANF 문제 정의

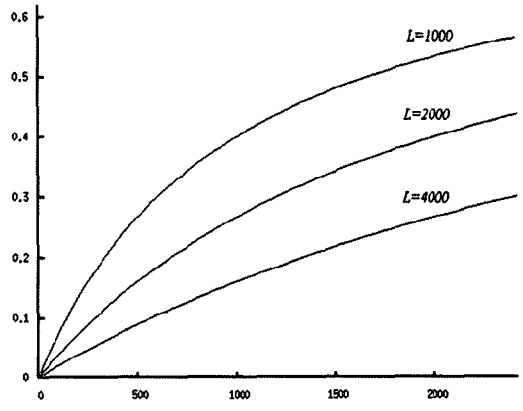
LANF 알고리즘은 기본적으로 NLCS 알고리즘에 기반하고 있다. NLCS와 LANF 문제는 같은 방식으로 정의되는데, $X[1..n]$, $Y[1..m]$간에 단위 길이당 LCS 유사도 점수 s/l 혹은 LANF 점수 $f(s)/g(l)$이 가장 높은 부분 문자열 $X[i'..i]$, $Y[j'..j]$를 찾는 것이다.

먼저 $X[i]$, $Y[j]$로 끝나는 부분 문자열 $X[i'..i]$, $Y[j'..j]$ 간의 LCS 집합 $L[i, j]$를 정의한다.

$$L[i, j] = \{\ X[i'..i]\text{와 } Y[j'..j]\text{ 간의 LCS},\ i' \le i \le n,\ j' \le j \le m\ \}$$



(a) k값이 비슷한 경우 NLA 점수



(b) 파라메터 L에 따른 NLA 점수

그림 1 크기 l에 대한 NLA 정규화 점수 $kl/(l+L)$

편의상 $L[i, j]$에 속하는 LCS를 유사도 점수 s와 시작 대각선 $d=i'+j'-2$ 쌍으로 대표한다.

$L[i, j] = \{ (s, d) \mid s = S(X[i'..i], Y[j'..j]),\ d=i'+j'-2 \}$

LCS 크기는 부분 문자열 $X[i'..i]$, $Y[j'..j]$ 길이의 합으로 정의되므로, 시작 대각선 $d=i'+j'-2$를 이용해 계산하면, $(i-i'+1) + (j-j'+1) = i+j - d$ 이다. 따라서 LCS (s, d)의 NLCS 점수는 $s/(i+j-d)$ 이다.

$L[i, j]$에 속하는 LCS 중에서 NLCS 점수 최대값을 $N[i, j]$로 정의한다.

$$N[i,j] = \max_{(s,d)\in L[i,j]} \frac{s}{i+j-d}$$

NLCS 문제는 모든 i, j에 대해 $N[i, j]$의 최대값을 찾는 문제가 된다. 마찬가지로 $L[i, j]$에 속하는 LCS에 대해 LANF 점수의 최대값 $N'[i, j]$는 다음과 같다.

$$N'[i,j] = \max_{(s,d)\in L[i,j]} \frac{f(s)}{g(i+j-d)}$$

여기서 f, g는 증가함수로 $a \le b \Rightarrow f(a) \le f(b)$ 조건을 만족하고, 추가로 local alignment 문제에서 s, l이 양수인 경우만 다루므로 $f(s) > 0$, $g(l) > 0$을 가정한다. LANF 문제도 모든 i, j에 대해 $N'[i, j]$의 최대값을 찾는 문제이다.

3.3 기존의 NLCS 알고리즘

LANF 알고리즘은 기존의 NLCS 알고리즘을 확장하여 얻는다. 먼저 간략하게 NLCS 알고리즘을 설명한다. NLCS 알고리즘은 두 문자열 간의 매치되는 문자만을 고려해 LCS를 계산한다. LCS에 대해서 두 문자열 길이의 곱 $O(nm)$ 크기의 테이블을 계산하는 대신, 매치되는 문자 개수를 r이라 하면 $r < nm$이고 $O(r\log\log n)$ 시간에 LCS를 계산할 수 있다[3,4]. NLCS 알고리즘은 이러한 매치되는 문자만을 고려하는 방법을 사용해 $O(rM\log\log n)$ 시간에 NLCS를 구한다[2]. 여기서 M은 유사도 점수 최대값인 가능한 최대 LCS 길이이다. 앞으로 문자열 간의 매치되는 문자 $X[i]=Y[j]$는 위치 i, j의 쌍 (i, j)로 나타낸다.

NLCS 알고리즘의 개관은 다음과 같다. NLCS 알고리즘은 각 매치 (i, j)를 i가 증가하는 순서, i가 같으면 j가 증가하는 순서로 처리한다. 매치 (i, j)를 덧붙여 확장할 수 있는 모든 LCS (s, d)를 유사도 점수 s별로 저장한 자료 구조를 유지한다. 이 자료 구조를 이용해 1부터 M까지 가능한 s에 대해 시작 대각선 d가 가장 큰 LCS들을 모으면, 이 중에서 $s/(i+j-d)$ 최대값을 구할 수 있다.

3.4 LCS 자료 구조

LANF 알고리즘에서 LCS를 저장하고 매치 (i, j)를 덧붙이는 방법은 NLCS와 같은 자료구조를 사용한다. 다음의 Lemma 2, 3은 매치되는 글자만을 고려한 LCS 알고리즘[4]의 기본 성질이며, NLCS에서도 이용되었다. 이들 Lemma로부터 같은 s값을 갖는 LCS중에서 시작 대각선 d가 가장 큰 LCS를 찾는 자료 구조를 얻을 수 있다.

어떤 LCS의 마지막 매치를 (i', j')라 하자. $X[i'..n]$, $Y[j'..m]$ 문자열간의 매치 중에서 (i', j')를 제외한 매치들을 해당 LCS에 덧붙이면 더 긴 LCS를 얻을 수 있다. 이러한 매치들의 집합을 $R(i', j')$라 하자

$$R(i', j') = \{ (i, j) \mid i' \le i \le n,\ j' \le j \le m \text{ 이고 } X[i]=Y[j] \} \setminus \{ (i', j') \}$$

Lemma 2 [4] 유사도 점수가 같은 두 LCS (s, d_1), (s, d_2)의 마지막 매치가 각각 (i_1, j_1), (i_2, j_2)라고 하자. LCS의 시작 대각선 d_1, d_2가 $d_1 \le d_2$이면, $R(i_1, j_1)$, $R(i_2, j_2)$의 교집합에 속하는 (i, j)에 대해 항상 $i+j-d_1 \ge i+j-d_2$ 이다.

위 Lemma 2에서 각 매치 (i, j)가 속할 R 집합은 유사도 점수 s가 같다면 시작 대각선 d값이 가장 큰 LCS에 의한 것 하나로 정해진다. $R(i_1, j_1)$, $R(i_2, j_2)$의 교집합에 속하는 매치들은 (i_2, j_2)에 붙여서 $s+1$길이 LCS로 확장하는 편이 alignment 크기가 작아서 NLCS 점수가 더 좋기 때문이다.

Lemma 3 [4] 유사도 점수가 같은 두 LCS (s, d_1), (s, d_2)의 마지막 매치가 각각 (i_1, j_1), (i_2, j_2)라고 하자. $d_1 \le d_2$인 $R(i_1, j_1)$, $R(i_2, j_2)$가 $i_1 \le i$, $i_2 \le i$ 인 (i, j)를 동시에 가질 수 있는 경우는 $j_1 < j_2$ 이다.

증명. LCS의 시작 대각선 d_1, d_2가 $d_1 \le d_2$ 라고 하자. j_1, j_2에 따라 두 가지 경우로 나눈다.

(1) $j_1 \ge j_2$: i가 i_1, i_2보다 크므로 (i, j)는 항상 $R(i_1, j_1)$, $R(i_2, j_2)$ 교집합에 속한다. 이 교집합은 시작 대각선 $d_1 \le d_2$ 인 $R(i_2, j_2)$에 속하므로 $R(i_1, j_1)$는 더 이상 매치 (i, j)를 포함할 수 없다.

(2) $j_1 < j_2$: $R(i_1, j_1)$는 $X[i_1..n]$, $Y[j_1..m]$간의 매치 중 $R(i_2, j_2)$에 속하는 매치를 제외한 $X[i_1..n]$, $Y[j_1..j_2]$간의 매치를 포함할 수 있다. □

Lemma 3에 따르면 유사도 점수가 같은 LCS를 저장하는 방법은 LCS의 마지막 매치 (i', j')의 Y위치 j'와 시작 대각선 d값이 증가하는 순서로 정렬해 두는 것이다. 이 정렬에서 벗어나는 LCS는 더 이상 확장할 매치를 가질 수 없는 LCS이다.

3.5 확장된 LANF 알고리즘

확장된 LANF 알고리즘은 증가함수 f, g를 도입해 Lemma 2를 확장한 Lemma 4에서 출발한다. 증가함수 f, g를 이용한 LANF 점수에 대해, s값이 같은 경우 $f(s)/g(i+j-d)$ 값을 결정하는 것은 시작 대각선 d이다. $f(s)$값이 같은 경우 NLCS 알고리즘과 마찬가지로 d값이 가장 큰 LCS를 찾으면 된다. 즉 Lemma 3에서 설

명한 LCS 저장 방법을 이용하면 NLCS와 마찬가지로 LANF를 구할 수 있다.

Lemma 4 유사도 점수가 같은 두 LCS (s, d_1), (s, d_2)의 마지막 매치가 각각 (i_1, j_1), (i_2, j_2)라고 하자. $d_1 \le d_2$ 이고 g가 증가함수이면, $R(i_1, j_1)$, $R(i_2, j_2)$의 교집합에 속하는 매치 (i, j)에 대해 항상 $g(i+j-d_1) \ge g(i+j-d_2)$ 이다.

증명. 주어진 LCS (s, d_1), (s, d_2)와 매치 (i, j)에 대해, g가 증가함수이고 $i+j-d_1 \ge i+j-d_2$ 이므로 $g(i+j-d_1) \ge g(i+j-d_2)$ 성립한다. □

LANF 알고리즘은 두 단계로 되어 있다. 첫번째 단계에서 LCS 자료 구조에서 매치 (i, j)를 붙일 수 있는 가장 큰 j'값을 갖는 LCS를 찾고 (i, j)를 덧붙이는 작업을 1부터 M까지 각 s에 대해 수행한다. 두번째 단계에서는 만들어진 s+1길이 LCS를 마지막 매치 j와 시작 대각선 d가 증가하는 순서가 되도록 LCS 자료 구조에 다시 저장한다.

그에 따라 LANF 알고리즘의 전체 pseudo 코드는 그림 2와 같다. 여기서 ROW는 $X[i]$와 매치되는 모든 $Y[j]$를 구해서 $X[i]$별로 매치 리스트를 $ROW[i]$에 저장해 둔 것이고, LRO[s]는 s길이 LCS가 마지막 매치 Y 위치 순서로 저장된 자료 구조를 나타낸다.

LANF 알고리즘이 걸리는 시간은 결국 매치 개수 r에 대해 M번 LCS가 저장된 자료구조를 탐색, 삽입, 삭제하는데 걸리는 시간이 된다. 기존의 NLCS 알고리즘과 마찬가지로 van Emde Boas 자료구조[5], Johnson Tree[6] 등의 자료구조를 이용해 모든 연산을 O(loglog$n$) 시간에 처리하여 전체 수행 시간은 $O(r$Mloglog$n)$ 시간이 된다[2].

4. 실험 결과

본 장에서는 실험을 통해 LANF 정규화 방법과 기존의 정규화 방법들을 비교한다. 실험을 통해 LANF 정규화 방법이 기존의 NLCS 방법을 개선하였고, NLA와 비교하여 인위적인 파라메터 L대신 미리 정의된 함수로 좋은 alignment를 찾을 수 있음을 보인다.

4.1 실험 방법

실험 방법은 먼저 Smith-Waterman 알고리즘을 사용하는 cross-match[7] 프로그램으로 내부에 유사도 점수가 낮은 부분을 포함하는 크기가 큰 alignment를 찾은 다음, 각각의 정규화 방법이 이러한 alignment를 어떻게 처리하는지 비교한다. 실험에 사용한 데이타는 NCBI GenBank[8]에서 사람과 쥐의 X염색체 중 비슷하다고 알려진 100k-300k 크기의 서열들이다(표 1).

각각의 정규화 방법을 비교하기 위해 답으로 내놓을 alignment 선택 기준을 통일한다. 실험에서 사용한 alignment 선택 기준은 매치되는 글자 50개 이상, 단위 길이당 매치되는 글자 비율 0.79이상으로 한다. 이는 사실상 NLCS 점수 기준이며, NLA 점수의 경우 파라메터 L에 따라 기준값을 정하기 애매하고 NLA 점수가 높으면 alignment 내부에 매치되는 글자 비율도 높기 때문에 같은 기준을 적용해 비교한다.

4.2 실험 결과 분석

표 1은 각각의 정규화 방법이 크기가 큰 alignment를 포함한 영역을 몇 개의 alignment로 구하는지 보여준다. LANF 정규화 방법은 NLCS가 잘게 쪼개는 alignment를 합쳐서 NLA와 비슷한 개수의 alignment를 구하며, NLA는 파라메터 L에 따라 구하는 alignment 개

```
1  for i ← 1 to n
2      s ← 0
3      // Construct s+1 LCS
4      do
5          for each match (i, j) in ROW[i]
6              j'max = max {j' | LCS with the last match (i', j') in LRO[s] and j' < j}
7              extend the LCS with the last match (i', j'max) to
                   a LCS with the last match (i, j)
8          s ← s+1
9      while LRO[s] is not empty
10
11     // Insert s+1 LCS to LRO[s+1]
12     max ← 0
13     while s > 0
14         for each match (i, j) in ROW[i]
15             LCS (s, d) ← the LCS with the last match (i, j) and the score s
16             if f(s)/g(i+j-d) > max then max ← f(s)/g(i+j-d)
17             insert LCS (s, d) into LRO[s]
18             for each (i', j') in LRO[s]
19                 LCS (s, d') ← the LCS with the last match (i', j') in LRO[s]
19                 if d' ≤ d and j' ≥ j then extract (s, d') from LRO[s]
20         s ← s-1
```

그림 2 LANF 알고리즘

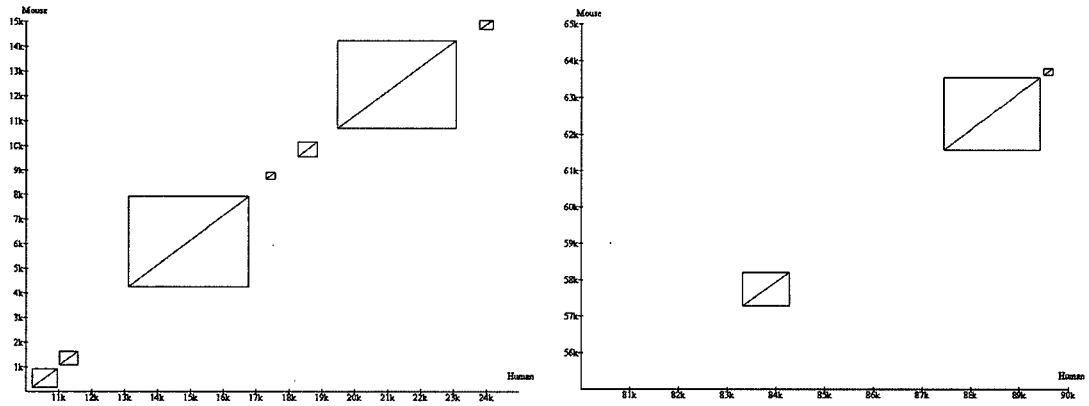
표 1 정규화 방법에 따른 alignment 개수

서열 Accession No. (Human / Mouse)	영역 (Human)	LANF	NLA				NLCS
			L=1000	L=2000	L=4000	L=8000	
AF030876 / AF121351	10k-25k	12	18	14	12	10	52
	80k-90k	6	8	8	8	8	21
AF030876 / AC106169	10k-17k	6	7	6	6	4	52
	19k-25k	4	6	5	5	3	51
	80k-90k	3	10	4	4	4	53
U52112 / AABR03119178	106k-110k	1	6	5	5	5	29
	123k-126k	4	6	6	4	4	27
U52111 / AC096338	70k-75k	6	10	8	8	7	36
BX936346 / AC094668	0k-8k	3	11	9	2	2	62
BX936365 / AC094668	0k-4k	1	8	7	5	4	25
	23k-29k	4	11	7	7	5	53

수가 달라진다. 각 실험들이 비슷한 경향을 보이기 때문에 여기서는 대표로 첫번째 실험 결과를 설명한다.

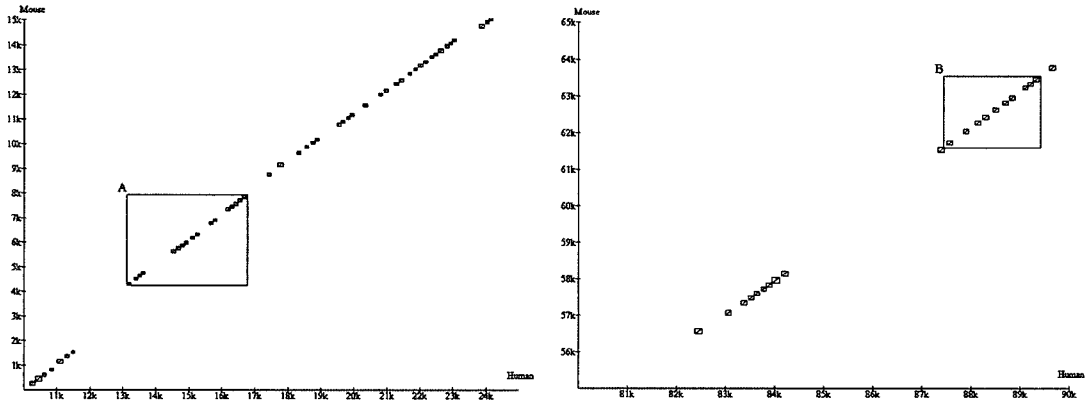
먼저 그림 3은 cross-match가 구한 크기가 큰 alignment 예시를 보여준다. 그림에서 각 X, Y축은 비교한 DNA 서열을 나타내고 alignment는 박스로 나타낸다.

다음의 그림 4는 기존의 NLCS 알고리즘이 크기가



(a) 사람 10-25k / 쥐 0-15k 영역 (b) 사람 80-90k / 쥐 55-65k 영역

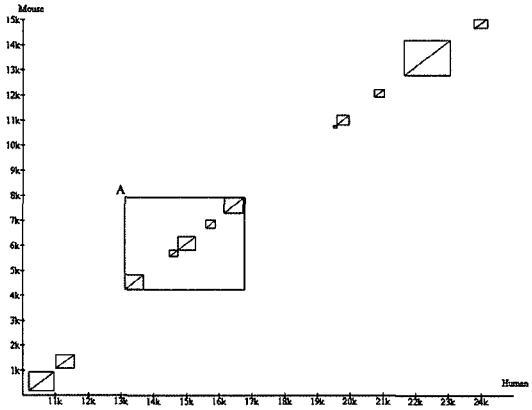
그림 3 cross-match를 이용한 사람/쥐 X염색체 비교 (GenBank accession No. AF030876, AF121351)



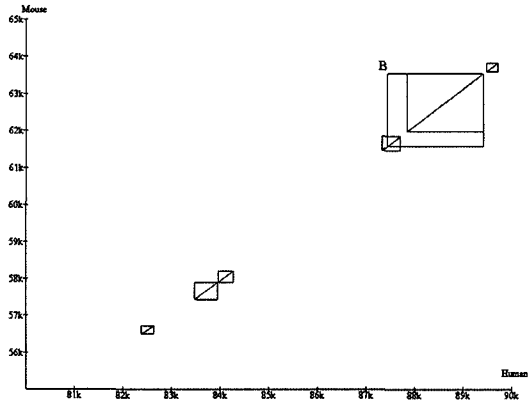
(a) 사람 10-25k / 쥐 0-15k 영역 (b) 사람 80-90k / 쥐 55-65k 영역

그림 4 NLCS를 이용한 사람/쥐 X염색체 비교

너무 작은 alignment를 구하는 것을 보여준다. 그림 4의 박스 A와 박스 B는 각각 그림 3의 cross-match가 구한 해당 영역의 alignment를 나타낸다. 박스 A내에 기존의 NLCS 알고리즘이 구한 alignment의 분포를 살펴보면, 영역 A는 3~4부분의 단위 길이당 유사도 점수가 높은 alignment로 이루어진 것을 알 수 있다. 그러

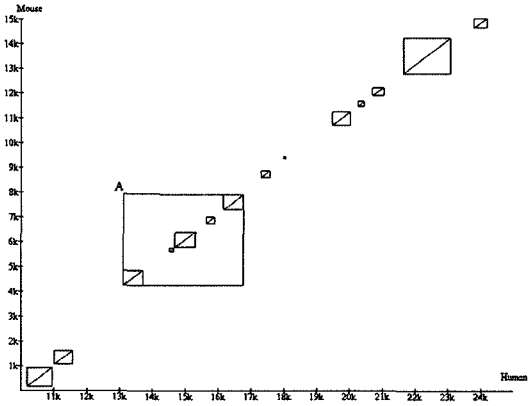


(a) 사람 10-25k / 쥐 0-15k 영역

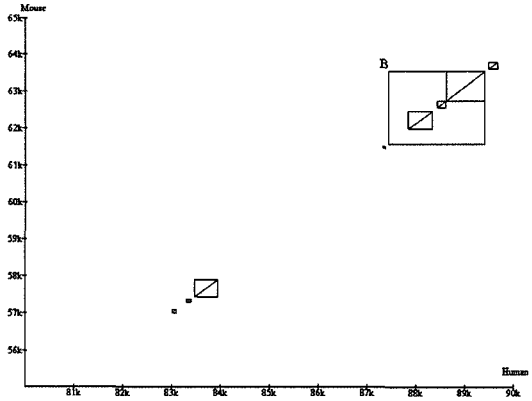


(b) 사람 80-90k / 쥐 55-65k 영역

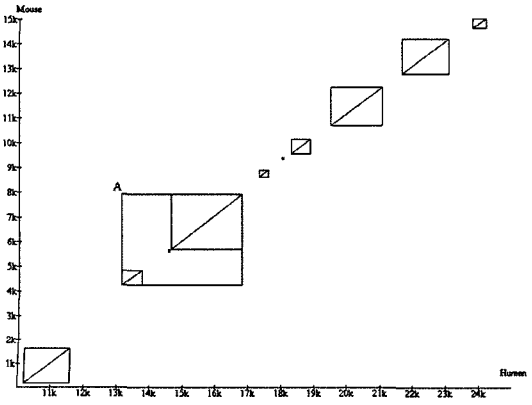
그림 5 LANF를 이용한 사람/쥐 X염색체 비교



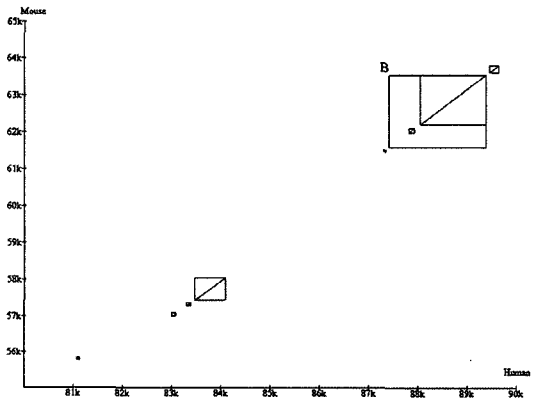
(a) 사람 10-25k / 쥐 0-15k 영역(L=2000)



(b) 사람 80-90k/ 쥐 55-65k 영역(L=1000)



(c) 사람 10-25k / 쥐 0-15k 영역(L=8000)



(d) 사람 80-90k / 쥐 55-65k 영역(L=2000)

그림 6 NLA를 이용한 사람/쥐 X염색체 비교

나 NLCS가 구한 alignment는 이보다 훨씬 많고 크기도 매우 작다. 영역 B는 매치가 골고루 분포해서 많아야 2~3부분으로 이루어진 alignment로 볼 수 있지만 NLCS 알고리즘은 이를 11개의 alignment로 쪼갠다.

다음의 그림 5는 LANF 알고리즘이 충분한 크기의 alignment를 구하는 것을 보여준다. 영역 A, B를 NLCS와 비교해 보면, NLCS에서 가까이 나타난 크기가 작은 alignment를 합쳐서 크기가 적당한 alignment를 구하는 것을 볼 수 있다.

다음의 그림 6은 NLA 결과가 파라메터 L에 따라 달라져서 실험에 공통으로 적용할 L을 정하는 것이 어렵다는 것을 보여준다. 영역 A와 영역 B를 동시에 잘 설명하는 결과는 파라메터 L을 2000정도로 잡았을 때이다 (그림 6(a), 그림 6(d)). 이보다 크면 그림 6(c)와 같이 너무 큰 alignment를 구하게 되고, 이보다 작으면 그림 6(b)와 같이 매치가 골고루 분포하는 alignment를 쪼개게 된다.

5. 결 론

본 논문에서는 기존의 NLCS 알고리즘을 수정해서 별다른 시간 손실 없이 더 좋은 alignment를 구하는 LANF 알고리즘을 제안했다. 여기서 좋은 alignment의 조건은 단위 길이당 유사도 점수가 높고 크기도 충분히 큰 것을 말한다. LANF 점수는 증가 함수 f, g를 도입하여 함수에 의해 정규화된 점수 $f(s)/g(l)$로 정의한다. 실험을 통해 $f(s)/g(l)$ 함수는 단위 길이당 유사도 점수 s/l이 같을 때 l에 대한 위로 볼록한 증가함수를 사용하는 것이 의미가 있음을 보였다.

참 고 문 헌

[1] A.N. Arslan, O. Egecioglu, and P.A. Pevzner, A new approach to sequence comparison: normalized sequence alignment, *Bioinformatics*, 17(4), 327-337, 2001.

[2] N. Efraty and G. M. Landau, Sparse normalized local alignment, In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern Matching (CPM)*, 333-346, 2004.

[3] D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano, Sparse dynamic programming I: linear cost functions, *Journal of ACM*, 39(3), 519-545, 1992.

[4] D. Gusfield, *Algorithms on strings, trees, and sequences*, Cambridge University Press, 1997.

[5] P. van Emde Boas, R. Kass, and E. Zijlstra, Design and implementation of an efficient priority queue, *Math. Systems Theory*, 10, 99-127, 1977.

[6] D.B. Johnson, A priority queue in which initialization and queue operations take $O(\log\log D)$ time, *Math. Systems Theory*, 15, 295-309, 1982.

[7] P. Green, http://www.phrap.org

[8] National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov

이 선 호

2002년 서울대학교 컴퓨터공학부 학사 2002년~현재 서울대학교 컴퓨터공학부 석박통합과정. 관심분야는 컴퓨터 이론, 알고리즘, 생물정보학.

박 근 수

1983년 서울대학교 컴퓨터공학과 학사 1985년 서울대학교 컴퓨터공학과 석사 1991년 미국 Columbia 대학교 전산학 박사. 1991년 11월~1993년 8월 영국 런던대학교 King's College 조교수. 1995년 7월~1995년 8월 호주 Curtin 대학교 방문연구원. 1993년 8월~현재 서울대학교 컴퓨터공학부 교수. 관심분야는 컴퓨터이론, 생물정보학, 암호학