N과 X를 포함하는 DNA 서열을 위한 효율적인 지역정렬 알고리즘

(An Efficient Local Alignment Algorithm for DNA Sequences including N and X)

김 진 욱 †

(Jin Wook Kim)

요 약 지역정렬(local alignment) 알고리즘은 주어진 두 서열에서 서로 유사한 부분 문자열을 찾아내는 알고리즘이다. DNA 서열은 A, C, G, T 외에 N과 X도 가질 수 있는데, N과 X는 DNA로부터 염기 배열 정보를 뽑아낼 때 실험적인 이유로 혹은 다른 이유로 일부 배열 정보를 잃어버린 경우에 사용된다. 본 논문에서는 A, C, G, T 이외에 N과 X를 모두 갖는 DNA 서열의 affine gap penalty metric에 대한 지역정렬을 찾는 효율적인 알고리즘을 제시한다. 이는 N만 처리할 수 있는 Kim-Park 알고리즘을 N과 X를 모두 처리할 수 있도록 성공적으로 확장한 결과이며, 더불어 새로운 문자가 추가되더라도 바로 적용이 가능한 일반화된 결과이다.

키워드 : DNA 서열, 지역정렬, affine gap penalty

Abstract A local alignment algorithm finds a substring pair of given two strings where two substrings of the pair are similar to each other. A DNA sequence can consist of not only A, C, G, and T but also N and X where N and X are used when the original bases lose their information for various reasons. In this paper, we present an efficient local alignment algorithm for two DNA sequences including N and X using the affine gap penalty metric. Our algorithm is an extended version of the Kim-Park algorithm and can be extended in case of including other characters which have similar properties to N and X.

Key words : DNA sequences, local alignment, affine gap penalty

1. 서 론

지역정렬(local alignment) 알고리즘은 주어진 두 서열에서 서로 유사한 부분 문자열을 찾아내는 알고리즘이다[1]. 지역정렬 알고리즘은 DNA 시퀀싱 프로그램에서 DNA 서열 조각들 간의 유사성을 확인하는 데 사용되며, 또한 서로 다른 종 간의 DNA 서열을 비교하는데 사용된다.

† 정 회 원 : 인하대학교 컴퓨터정보공학부 교수
gnugi@inha.ac.kr
논문접수 : 2009년 11월 30일
심사완료 : 2010년 1월 11일

정보과학회논문지: 컴퓨팅의 실제 및 레터 제16권 제3호(2010.3)

DNA 서열은 실제 DNA의 염기 배열을 사람 혹은 컴퓨터가 쉽게 이해할 수 있도록 A, C, G, T의 네 가지 문자를 사용하여 문자열로 표현한 것이다. 그런데, DNA로부터 염기 배열 정보를 뽑아낼 때 실험적인 이유로 혹은 다른 이유로 일부 배열 정보를 잃어버릴 수가 있다[2-6]. 이렇게 배열 정보를 알 수 없는 부분을 표현하기 위해 A, C, G, T 외에 N과 X가 사용된다.

N과 X가 동시에 사용되는 경우 이 둘은 서로 다른 의미를 가지게 된다[2]. DNA 시퀀싱 프로그램인 PHRAP에서 N은 실험적으로 어떤 염기인지 확인할 수 없는 경우를 표현하고, X는 염기 정보가 필요없는 경우를 표현한다. 따라서 지역정렬을 구하는 데 사용되는 유사도 점수로 N은 일치도 불일치도 아닌 그 사이 값을, X는 불일치보다 더 작은 값을 사용한다(그림 1 참고).

지역정렬을 구하는 이전 연구 결과들은 다음과 같다. 주어진 두 서열의 길이가 n과 m일 때, Smith-Waterman 알고리즘[7]은 affine gap penalty metric에

	A	C	G	T	N	X
A	1	-2	-2	-2	0	-3
C	-2	1	-2	-2	0	-3
G	-2	-2	1	-2	0	-3
T	-2	-2	-2	1	0	-3
N	0	0	0	0	0	0
X	-3	-3	-3	-3	0	-3

그림 1 DNA 서열에 대한 점수 행렬의 한 예. 여기에서는 $\delta=2$, $\tau=-3$, $\sigma=0$인 경우임

대한 지역정렬을 $O(n^2m)$ 시간에 구할 수 있고, Gotoh 알고리즘[8]은 이를 $O(nm)$ 시간으로 개선하였다. 주어진 두 서열에서 N의 개수가 T'과 S'이고 연속된 N들의 개수가 v'와 w'일 때, Kim-Park 알고리즘[9]은 Gotoh의 알고리즘을 $O((n-T')(m-S')+v'm+w'n)$ 시간으로 개선하였다. 하지만, Kim-Park 알고리즘에서는 N과 X를 모두 N으로 간주하고 문제를 풀고 있으며, N과 X를 구분하는 경우에는 바로 적용할 수가 없다.

이에 본 논문에서는 A, C, G, T 이외에 N과 X를 모두 갖는 DNA 서열의 affine gap penalty metric에 대한 지역정렬을 찾는 $O((n-T)(m-S)+vm+wn)$ 시간 알고리즘을 제시한다. 여기서 T와 S는 주어진 두 서열에 있는 N과 X의 전체 개수이고 v와 w는 두 서열에 있는 연속된 N들과 연속된 X들의 개수의 합이다. 이는 이전 연구 결과[9]를 N과 X를 모두 처리할 수 있도록 성공적으로 확장한 결과이며, 더불어 새로운 문자가 추가되더라도 바로 적용이 가능한 일반화된 결과이다.

본 논문의 구성은 다음과 같다. 우선 2장에서는 기본적인 용어 및 관련 연구를 살펴보고, 3장에서 본 논문에서 제시하는 알고리즘을 설명한다. 4장에서 복잡도를 분석한 뒤 5장에서 결론을 맺는다.

2. 관련 연구

먼저 몇 가지 용어들을 정의한다. 문자열 혹은 서열은 알파벳 Σ에 속하는 문자들이 0개 이상 연결된 것을 말한다. 공백은 Δ로 표현하고 Σ에 속하지 않지만 편의상 문자로 간주한다. 서열 A의 길이는 $|A|$로 표기한다. a_i는 서열 A의 i번째 문자를 표시하며, $A[i..j]$는 A의 부분 문자열 $a_ia_{i+1}...a_j$를 의미한다. $\alpha<A$는 α가 A의 부분 문자열임을 표시한다. 두 개의 서열 $A=a_1a_2...a_n$과 $B=b_1b_2...b_m$이 주어질 때, A와 B의 정렬(alignment)은 A와 B에 0개 이상의 공백을 추가한 $A^*=a_1^*a_2^*...a_l^*$와 $B^*=b_1^*b_2^*...b_l^*$가 되며 $0\le i\le l$인 i에 대해 a_i^*와 b_i^*는 서로 대응된다. 이때 다음과 같은 세 가지 대응관계가 존재한다.

- 일치: $a_i^*=b_i^*\neq\Delta$,
- 불일치: $a_i^*\neq b_i^*$이며 $a_i^*, b_i^*\neq\Delta$,
- 삽입 혹은 삭제: a_i^*와 b_i^* 중 하나만 Δ.

참고로, $a_i^*=b_i^*=\Delta$인 경우는 없다.

2.1 지역정렬(Local alignment)

두 개의 서열 A와 B가 주어질 때, A와 B의 지역정렬은 $\alpha<A$이고 $\beta<B$인 α, β의 정렬로 정의되고, A와 B의 최적의 지역정렬은 A와 B의 지역정렬들 중 가장 높은 유사도를 갖는 지역정렬로 정의된다. 앞으로 $SL(A,B)$는 A와 B의 최적의 지역정렬의 유사도를 나타낸다.

최적의 지역정렬을 구하는 대표적인 알고리즘으로 Smith-Waterman 알고리즘[7]과 Gotoh 알고리즘[8]이 있다(합쳐서 SWG 알고리즘으로 표현하겠다). $|A|=n$, $|B|=m$인 두 서열 A, B가 주어지면, SWG 알고리즘은 크기$(n+1)(m+1)$인 동적 프로그래밍 테이블 H table을 이용하여 $SL(A,B)$를 계산한다. $0\le i\le n$, $0\le j\le m$인 i, j에 대해 H_{ij}는 다음의 점화 관계를 통해 계산된다.

$$H_{i,0}=H_{0,j}=0\ \ (0\le i\le n,\ 0\le j\le m)$$

$$H_{ij}=\max\begin{Bmatrix}H_{i-1,j-1}+s(a_i,b_j),\\ C_{ij},R_{ij},0\end{Bmatrix}\ \ \begin{matrix}(1\le i\le n,\\ 1\le j\le m)\end{matrix}$$

이때, $s(a_i,b_j)$, C_{ij}, R_{ij}는 다음과 같이 정의된다.

- $s(a_i,b_j)$는 a_i와 b_j의 유사도 점수를 나타낸다.

$$s(a_i,b_j)=\begin{cases}1 & a_i=b_j\text{인 경우}\\ -\delta & a_i\neq b_j\text{인 경우}\end{cases}$$

단, δ는 0 이상인 상수이다.

- C_{ij}는 a_i의 대응관계가 삽입인 $A[1..i]$와 $B[1..j]$의 지역정렬 중 가장 유사도가 높은 것을 나타낸다.

$$C_{0,j}=-\infty\ \ (1\le j\le m)$$

$$C_{ij}=\max\begin{Bmatrix}H_{i-1,j}-\gamma-\mu,\\ C_{i-1,j}-\mu\end{Bmatrix}\ \ \begin{matrix}(1\le i\le n,\\ 1\le j\le m)\end{matrix}$$

단, γ와 μ는 0 이상인 상수이다.

- R_{ij}는 b_j의 대응관계가 삭제인 $A[1..i]$와 $B[1..j]$의 지역정렬 중 가장 유사도가 높은 것을 나타낸다.

$$R_{i,0}=-\infty\ \ (1\le i\le n)$$

$$R_{ij}=\max\begin{Bmatrix}H_{i,j-1}-\gamma-\mu,\\ R_{i,j-1}-\mu\end{Bmatrix}\ \ \begin{matrix}(1\le i\le n,\\ 1\le j\le m)\end{matrix}$$

결국 모든 H_{ij} 중 가장 점수가 높은 것이 $SL(A,B)$가 되고 이는 $O(mn)$ 시간에 계산이 된다.

2.2 Affine gap penalty metric[1]

삽입이나 삭제의 경우 유사도 계산에 gap penalty가 적용된다. k개의 연속된 삽입(혹은 삭제)에 대해 affine gap penalty g_k는 다음과 같이 정의된다.

$$g_k=\gamma+k\mu$$

이때, γ와 μ는 0 이상인 상수로, γ를 gap 초기화 penalty, μ를 gap 확장 penalty라고 한다.

2.3 점수 행렬(Scoring matrix)

DNA 서열은 A, C, G, T의 네 가지 문자로 구성되지만, 확정되지 않은 염기를 표현하기 위한 다른 문자들도 존재한다[10]. 그 중 N과 X가 DNA 시퀀싱 프로그램에서 사용된다[2]. N은 'any' 혹은 'unspecified'를 의미하고 실험적으로 어떤 염기인지 확인할 수 없을 때 사용되며, X는 'unknown'을 의미하고 vector screening 또는 repeat masking된 위치에 사용된다.

그림 1은 일반적으로 사용되는 DNA 서열에 대한 유사도 점수('점수 행렬'로도 불린다.)의 한 예이다. A, C, G, T의 경우에는 일치인 경우에는 1을, 불일치인 경우에는 -2를 사용하지만, N이나 X인 경우에는 일치, 불일치의 의미가 없이 한 문자라도 N이면 항상 0을, 한 문자가 X이고 다른 문자가 N이 아니라면 항상 -3을 사용하고 있다.

이에 우리는 앞에서 정의한 유사도 점수 $s(a_i,b_j)$를 다음과 같이 확장할 수 있다.

$$s(a_i,b_j)=\begin{cases}1 & a_i=b_j\neq \text{X,N인 경우}\\ -\delta & a_i\neq b_j\text{이며 } a_i,b_j\neq \text{X,N인 경우}\\ \tau & a_i\text{나 } b_j\text{가 X이고 } a_i,b_j\neq \text{N인 경우}\\ \sigma & a_i\text{나 } b_j\text{가 N인 경우}\end{cases}$$

단, τ와 σ는 임의의 상수이다. 앞의 예에서는 $\delta=2$, $\tau=-3$, $\sigma=0$인 경우가 된다. 그림 2는 H table에서 각각의 유사도 점수가 사용되는 영역을 나타낸 그림이다.

그림 2에서 나눠진 각각의 직사각형을 블록이라고 부른다. 즉, $A[i_1..i_2]$가 X만 있는 부분이거나 N만 있는 부분이거나 혹은 X와 N이 없는 부분이고, $B[j_1..j_2]$가 X만 있는 부분이거나 N만 있는 부분이거나 혹은 X와 N이 없는 부분일 때, 부분행렬 $H_{i_1..i_2,j_1..j_2}$가 하나의 블록이 된다.

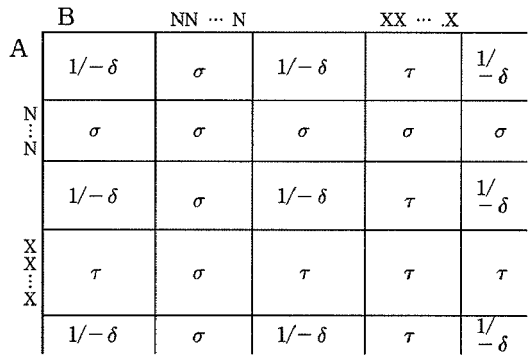

그림 2 N과 X를 갖는 두 서열에 대한 H table에서 각각의 유사도 점수가 사용되는 영역

본 논문에서 다룰 문제는 다음과 같다.

문제 1. A, C, G, T 및 N과 X로 구성된 두 개의 DNA 서열 A, B가 주어졌을 때, affine gap penalty metric에 대한 $SL(A,B)$를 구하라.

3. 알고리즘

$|A|=n$, $|B|=m$인 두 서열 A, B가 주어질 때, $SL(A,B)$를 구하기 위해 H table을 계산하는데, 이때 H table을 다음과 같이 다섯 가지 경우의 블록으로 나누어 계산한다.

경우 1: a_i와 b_j가 X도 N도 아닌 경우. 블록을 모두 계산.

경우 2: a_i와 b_j 중 하나는 X이고 다른 하나는 N이 아닌 경우. 블록의 가장 아래 행과 가장 오른쪽 열만 계산.

경우 3: a_i만 N인 경우. 블록의 가장 아래 행만 계산.

경우 4: b_j만 N인 경우. 블록의 가장 오른쪽 열만 계산.

경우 5: a_i와 b_j가 모두 N인 경우. 블록의 왼쪽에 인접한 열과 위쪽에 인접한 행 및 해당 블록의 가장 오른쪽 아래 모퉁이 하나만 계산.

그림 3은 다섯 가지 경우로 나뉜 H table의 한 예를 보여준다. 경우 1은 유사도 점수로 $1/-\delta$가 사용되는 영역으로 SWG 알고리즘[7,8]으로 계산할 수 있다. 경우 2는 τ가 사용되는 영역으로 3.1절에서 설명한다. 경우 3, 4, 5는 σ가 사용되는 영역으로 Kim-Park 알고리즘[9]으로 계산할 수 있다.

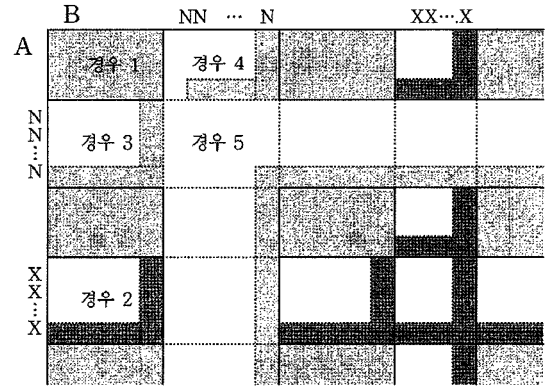

그림 3 다섯 가지 경우의 블록으로 표현된 H table. $SL(A,B)$를 구하려면 색칠된 부분만 계산하면 됨. 경우 2가 X와 관련된 블록들임

3.1 경우 2

경우 2는 a_i와 b_j 중 하나는 X이고 다른 하나는 N이 아닌 경우로, 해당 블록의 가장 아래 행과 가장 오른쪽

열만 계산하면 된다. 즉, $H_{i-p+1..i,j-q+1..j}$가 경우 2인 블록일 때, $C_{i-k,j-q}$, $R_{i-k,j-q}$, $H_{i-k,j-q}(0 \le k \le p)$와 $C_{i-p,j-l}$, $R_{i-p,j-l}$, $H_{i-p,j-l}(0 \le l \le q)$을 이용하여 $C_{i,j-l}$, $R_{i,j-l}$, $H_{i,j-l}(0 \le l < q)$과 $C_{i-k,j}$, $R_{i-k,j}$, $H_{i-k,j}(0 \le k < p)$를 계산한다.

이 방식은 run-length encoded 문자열의 유사도 계산 방식[11]을 원용한 것으로, 이렇게 하나의 행과 하나의 열만 계산해도 되는 이유는 해당 블록의 가장 아래 행이 계산되면 그 아래의 블록의 계산에 필요한 입력을 줄 수 있고, 해당 블록의 가장 오른쪽 열이 계산되면 그 오른쪽의 블록의 계산에 필요한 입력을 줄 수 있기 때문이다.

지금부터 경우 2인 블록에서 가장 아래 행을 계산할 수 있는 점화식을 제시한다. 참고로, 이 점화식의 행과 열을 대칭적으로 변환하면 가장 오른쪽 행을 계산할 수 있는 점화식을 바로 얻을 수 있다.

3.1.1 $C_{i,j-l}$ 계산

$C_{i,j-l}(0 \le l < q)$을 계산하기 위해서는 기본적으로 $R_{i-k,j-q}$, $H_{i-k,j-q}(1 \le k < p)$와 $C_{i-p,j-s}$, $H_{i-p,j-s}$ $(l \le s \le q)$가 필요하다. 이들로부터 $C_{i,j-l}$을 계산하기 위해서는 다양한 경로를 고려해야 하는데, 다음 보조정리들을 통하여 그 중 꼭 필요한 경로만을 찾도록 한다. 이러한 경로를 필수경로라 부른다[11].

먼저 $C_{i-p,j-s}$와 $H_{i-p,j-s}(l \le s \le q)$를 살펴본다. 보조정리 1과 2는 [9]에 증명되어 있다.

보조정리 1. [9]$H_{i-p,j-u}(l \le u \le q)$로부터의 경로로 계산되는 $C_{i,j-l}$보다 $H_{i-p,j-s}(l \le s \le \min\{l+p-1,q\})$로부터의 경로로 계산되는 $C_{i,j-l}$이 더 크거나 같은 $H_{i-p,j-s}$가 적어도 하나 존재한다(그림 4(a) 참고).

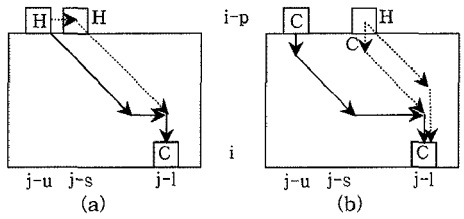

그림 4 (a) 보조정리 1 (b) 보조정리 2

보조정리 2. [9]$C_{i-p,j-u}(l < u < q)$로부터의 경로로 계산되는 $C_{i,j-l}$보다 $H_{i-p,j-s}(l \le s \le \min\{l+p-2,q\})$로부터의 경로로 계산되는 $C_{i,j-l}$이 더 크거나 같은 $H_{i-p,j-s}$가 적어도 하나 존재한다(그림 4(b) 참고).

보조정리 1과 2에서 찾은 필수경로는 $\max_{l \le s \le \min\{l+p-1,q\}}\{H_{i-p,j-s}+(s-l)\tau-g_{p-(s-l)}\}$이고 다루지 않은 필수경로는 $C_{i-p,j-l}-p\mu$이다.

다음은 $R_{i-k,j-q}$와 $H_{i-k,j-q}(1 \le k < p)$를 살펴본다.

보조정리 3. 만약 $\tau > -2\mu$라면, $R_{i-u,j-q}(q-l < u < p)$로부터의 경로로 계산되는 $C_{i,j-l}$보다 $H_{i-u,j-q}$로부터의 경로로 계산되는 $C_{i,j-q+l}$이 더 크다(그림 5 참고).

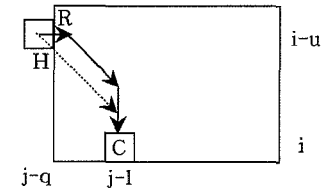

그림 5 보조정리 3

증명. $\tau > -2\mu$일 때 $R_{i-u,j-q}(q-l < u < p)$로부터 $C_{i,j-l}$까지 최대값을 주는 경로는 $R_{i-u,j-q}-\mu+(q-l-1)\tau-g_{u-q+l+1}$이다. 그런데, H table의 정의에 의해 $R_{i-u,j-q} \le H_{i-u,j-q}$이다. 그리고 $-2\mu < \tau$이므로,

$$\begin{aligned} R_{i-u,j-q}-\mu+(q-l-1)\tau-g_{u-q+l+1} &\le H_{i-u,j-q}-\mu+(q-l-1)\tau-g_{u-q+l+1} \\ &< H_{i-u,j-q}+(q-l)\tau-g_{u-q+l} \end{aligned}$$

이 된다. 즉, $H_{i-u,j-q}$로부터의 경로로 계산되는 $C_{i,j-q+l}$이 더 크다. □

$\tau > -2\mu$인 경우에 보조정리 3에서 다루지 않은 필수경로는 $\max_{1 < s \le \min\{q-l,p-1\}}\{R_{i-s,j-q}-(q-l-s+1)\mu+(s-1)\tau\}-g_1$이고, 만약 $\tau \le -2\mu$라면 필수경로는 $\max_{1 \le s < p}\{R_{i-s,j-q}-g_s\}-(q-l)\mu$이다. $H_{i-k,j-q}$ $(1 \le k < p)$는 모두 필수경로에 관여한다. 이상으로부터 도출한 $C_{i,j-l}$을 위한 점화식은 다음과 같다(그림 6 참고).

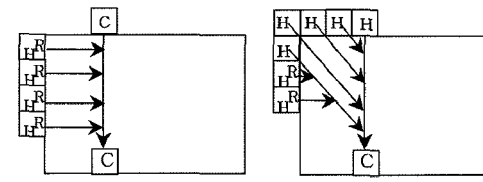

그림 6 $C_{i,j-l}$ 계산을 위한 필수경로들

$$C_{i,j-l} = \max\begin{cases} C_{i-p,j-l}-p\mu \\ \max_{l \le s \le \min\{l+p-1,q\}}\{H_{i-p,j-s}+(s-l)\tau-g_{p-(s-l)}\} \\ \max_{1 \le s < p}\{R_{i-s,j-q}-g_s\}-(q-l)\mu \\ \max_{1 \le s < p}\{H_{i-s,j-q}-g_s\}-g_{q-l} \\ \max_{1 < s \le \min\{q-l,p-1\}}\{R_{i-s,j-q}-(q-l-s+1)\mu+(s-1)\tau\}-g_1 \\ \max_{1 < s \le \min\{q-l,p-1\}}\{H_{i-s,j-q}-g_{q-l-s+1}+(s-1)\tau\}-g_1 \\ \max_{q-l < s \le p-1}\{H_{i-s,j-q}-g_{s-q+l}\}+(q-l)\tau \end{cases}$$

$C_{i,j-l}$ 하나의 값을 계산하기 위해서는 $O(p)$의 시간이 필요하지만, $0 \le l < q$인 l 전체에 대해서는 $O(p+q)$ 시간이면 충분하다. 즉, 처음 $C_{i,j-q+1}$을 계산하는 데에는 $O(p)$ 시간이 걸리지만, 이후 l을 $q-2$부터 0까지 변화시키며 $C_{i,j-l}$을 계산할 때 하나의 $C_{i,j-l}$은 $O(1)$ 시간에 계산이 가능하다. 이는 인접한 두 값 $C_{i,j-l-1}$과 $C_{i,j-l}$의 계산식이 유사하다는 점과 자료구조 MQUEUE에 바탕을 둔다[9,11].

3.1.2 $R_{i,j-l}$ 계산[11]

$R_{i,j-l}(0 \le l < q)$을 계산하기 위해서는 $R_{i,j-l-1}$과 $H_{i,j-l-1}$이 필요하다. 그런데, $R_{i,j-q}$와 $H_{i,j-q}$를 입력으로 알고 있고 블록의 가장 아래 행을 계산할 때 왼쪽에서 오른쪽으로 계산한다면, $R_{i,j-l}$을 계산하는 시점에 $R_{i,j-l-1}$과 $H_{i,j-l-1}$을 모두 알고 있으므로 상수시간에 계산이 가능하다.

3.1.3 $H_{i,j-l}$ 계산

$H_{i,j-l}(0 \le l < q)$을 계산하기 위해서는 $C_{i,j-l}$, $R_{i,j-l}$, 그리고 $H_{i-1,j-l-1}$이 필요하다. $C_{i,j-l}$과 $R_{i,j-l}$은 앞에서 구했으므로, 다음 보조정리들을 통하여 $H_{i-1,j-l-1}$을 거쳐서 τ로 $H_{i,j-l}$에 도달하는 필수경로들을 찾도록 한다.

먼저 $H_{i-k,j-q}(1 \le k < p)$와 $H_{i-p,j-s}(l < s \le q)$를 살펴본다. 보조정리 4와 5는 [11]에 증명되어 있다.

보조정리 4. [11]$H_{i-u,j-q}(l \le u < p)$로부터 $H_{i-1,j-l-1}$을 거쳐서 마지막이 τ인 경로로 계산되는 $H_{i,j-l}$보다 $R_{i,j-l}$이나 $H_{i-q+l,j-q}+(q-l)\tau$(단, $p > q-l$인 경우)로 계산되는 $H_{i,j-l}$이 더 크거나 같다.

보조정리 5. [11]$H_{i-p,j-u}(l \le u \le q)$로부터 $H_{i-1,j-l-1}$을 거쳐서 마지막이 τ인 경로로 계산되는 $H_{i,j-l}$보다 $C_{i,j-l}$이나 $H_{i-p,j-l-p}+p\tau$(단, $p \le q-l$인 경우)로 계산되는 $H_{i,j-l}$이 더 크거나 같다.

보조정리 4와 5로 찾은 필수경로는 $H_{i-q+l,j-q}+(q-l)\tau$(단, $p > q-l$인 경우) 혹은 $H_{i-p,j-l-p}+p\tau$(단, $p \le q-l$인 경우)이다.

다음은 $R_{i-k,j-q}(1 \le k < p)$와 $C_{i-p,j-s}(l < s < q)$를 살펴본다. 보조정리 6은 [9]에 증명되어 있고, 보조정리 7은 보조정리 6과 대칭적인 관계이므로 증명은 생략한다.

보조정리 6. [9]$C_{i-p,j-u}(l < u < q)$로부터의 경로로 계산되는 $H_{i-1,j-l-1}$보다 $C_{i-p,j-s}(l < s < \min\{l+p,q\})$로부터의 경로로 계산되는 $H_{i-1,j-l-1}$이 더 크거나 같은 $C_{i-p,j-s}$가 적어도 하나 존재한다.

보조정리 7. $R_{i-u,j-q}(1 \le u < p)$로부터의 경로로 계산되는 $H_{i-1,j-l-1}$보다 $R_{i-s,j-q}(1 \le s < \min\{q-l,p\})$로부터의 경로로 계산되는 $H_{i-1,j-l-1}$이 더 크거나 같은 $R_{i-s,j-q}$가 적어도 하나 존재한다.

보조정리 6과 7로 찾은 필수경로는 $\max_{l < s < \min\{l+p,q\}}\{C_{i-p,j-s}-(p-s+l)\mu+(s-l)\tau\}$와 $\max_{1 \le s < \min\{q-l,p\}}\{R_{i-s,j-q}-(q-l-s)\mu+s\tau\}$이다. 이상으로부터 도출한 $H_{i,j-l}$을 위한 점화식은 다음과 같다(그림 7 참고).

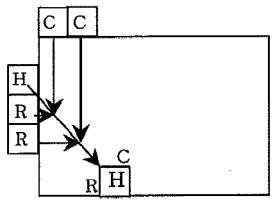

그림 7 $H_{i,j-l}$ 계산을 위한 필수경로들

$$H_{i,j-l} = \max\begin{cases} C_{i,j-l} \\ R_{i,j-l} \\ H_{i-s,j-l-s}+s\tau \quad (\text{단, } s=\min\{p,q-l\}) \\ \max_{l < s < \min\{l+p,q\}}\{C_{i-p,j-s}-(p-s+l)\mu+(s-l)\tau\} \\ \max_{1 \le s < \min\{q-l,p\}}\{R_{i-s,j-q}-(q-l-s)\mu+s\tau\} \end{cases}$$

$H_{i,j-l}$ 하나의 값을 계산하기 위해서는 $O(p)$의 시간이 필요하지만, $0 \le l < q$인 l 전체에 대해서는 $C_{i,j-l}$의 계산과 마찬가지로 $O(p+q)$의 시간에 수행된다.

4. 시간 복잡도

이번 장에서는 본 논문에서 제시하는 알고리즘의 시간 복잡도를 알아본다. $|A|=n$, $|B|=m$인 두 서열 A, B가 주어질 때, T와 S는 서열 A와 B에 있는 X와 N의 전체 개수를 나타내고, v와 w는 서열 A와 B에 있는 연속된 X와 연속된 N의 영역의 개수의 합을 나타낸다. 또한, N 영역의 좌우에는 최대 2개의 N이 아닌 영역이 올 수 있고 X 영역의 좌우에는 최대 2개의 X가 아닌 영역이 올 수 있으므로, 서열 A와 B는 최대 $3v$개와 $3w$개의 영역을 가진다.

먼저 크기가 $p \times q$인 한 블록을 계산하는데 필요한 시간을 경우별로 살펴보면 다음과 같다. 경우 1은 크기가 $p \times q$인 블록에 대해 $O(pq)$의 시간이 걸리고[8], 경우 2는 $O(p+q)$의 시간이 걸린다. 경우 3은 $O(q)$, 경우 4는 $O(p)$, 마지막으로 경우 5는 $O(\min\{p,q\})$의 시간이 걸린다[9]. 그런데, 계산의 편의를 위해 경우 2부터 5까지 모두 $O(p+q)$로 간주하겠다.

전체에 대한 시간 복잡도는 다음과 같다. 경우 1인 블

록을 모두 모으면 크기가 $(n-T)(m-S)$가 되고 따라서 $O((n-T)(m-S))$의 시간이 필요하다. 경우 2부터 5까지를 살펴보면, A에는 길이의 합이 T인 v개의 X 혹은 N 영역이 있고 B에는 길이의 합이 m인 최대 $3w$개의 영역이 있으므로, 이들이 만드는 블록들의 행의 합은 vm이며 열의 합은 $3wT$이다. 반대로, B에는 길이의 합이 S인 w개의 X 혹은 N 영역이 있고 A에는 길이의 합이 n인 최대 $3v$개의 영역이 있으므로, 이들이 만드는 블록들의 행의 합은 최대 $3vS$이며 열의 합은 최대 wn이다. 이때, A와 B 모두 X 혹은 N인 영역이 중복되어 계산되었고 이들이 만드는 블록들의 크기는 행의 합이 vS, 열의 합이 wT이다. 따라서 전체 행의 크기는 최대 $vm+3vS-vS=vm+2vS\leq 3vm$이 되고 열의 크기는 같은 식으로 최대 $3wn$이 되어, 경우 2, 3, 4, 5에 대한 전체 시간 복잡도는 $O(vm+wn)$이 된다. 결국 전체 알고리즘의 시간 복잡도는 $O((n-T)(m-S)+vm+wn)$이 된다.

정리 1. N과 X를 포함하는 두 DNA 서열의 affine gap penalty metric에 대한 최적의 지역정렬은 $O((n-T)(m-S)+vm+wn)$의 시간에 구해진다.

5. 결 론

본 논문에서는 A, C, G, T 이외에 N과 X를 모두 갖는 DNA 서열의 affine gap penalty metric에 대한 최적의 지역정렬을 찾는 효율적인 알고리즘을 제시하였다. 이는 X가 포함된 영역을 효율적으로 처리할 수 있는 알고리즘을 N만 처리하는 Kim-Park 알고리즘에 적용하여 N과 X를 모두 처리할 수 있도록 한 것이다.

또한, X가 포함된 영역을 다루는 알고리즘은 새로운 문자가 추가되어 새로운 유사도 점수가 생기더라도 그에 해당하는 블록에 바로 적용이 가능하다. 왜냐하면 X의 유사도 점수 τ를 일치 및 불일치의 유사도 점수와는 무관하게 일반적인 경우로 두고 알고리즘을 만들었기 때문이다.

참 고 문 헌

[1] D. Gusfield, *Algorithms on Strings, Trees, and Sequences*, Cambridge University Press, New York, 1997.

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

[3] E.W. Myers, G.G. Sutton, A.L. Delcher, I.M. Dew, D.P. Fasulo, et al., A Whole-Genome Assembly of Drosophila, *Science*, 287, pp.2196-2204, 2000.

[4] A. Batzoglou, D.B. Jaffe, K. Stanley, J. Butler, et al., ARACHNE: A Whole-Genome Shotgun Assembler, *Genome Research*, 12, pp.177-189, 2002.

[5] J. Wang, G.K. Wong, P. Ni, et al., RePS: A Sequence Assembler that Masks Exact Repeats Identified from the Shotgun Data, *Genome Research*, 12, pp.824-831, 2002.

[6] J.W. Kim, K. Roh, K. Park, H. Park, J. Seo, MLP: Mate-Based Sequence Layout with PHRAP, *Bioinformatics and Biosystems*, 1(1), pp.61-66, 2006.

[7] T.F. Smith, M.S. Waterman, Identification of Common Molecular Subsequences, *Journal of Molecular Biology*, 147, pp.195-197, 1981.

[8] O. Gotoh, An Improved Algorithm for Matching Biological Sequences, *Journal of Molecular Biology*, 162, pp.705-708, 1982.

[9] J.W. Kim, K. Park, An Efficient Alignment Algorithm for Masked Sequences, *Theoretical Computer Science*, 370, pp.19-33, 2007.

[10] NC-UIB, Nomenclature for Incompletely Specified Bases in Nucleic Acid Sequences. Recommendations 1984, *The European Journal of Biochemistry*, 150, pp.1-5, 1985.

[11] J.W. Kim, A. Amir, G.M. Landau, K. Park, Computing Similarity of Run-Length Encoded Strings with Affine Gap Penalty, *Theoretical Computer Science*, 395, pp.268-282, 2008.

김 진 욱

1998년 서울대학교 수학과 학사. 2000년 서울대학교 컴퓨터공학과 석사. 2006년 서울대학교 전기·컴퓨터공학부 박사. 2006년~2009년 ㈜에이치엠연구소 책임연구원. 2009년~현재 인하대학교 컴퓨터정보공학부 연구교수. 관심분야는 컴퓨터이론, 알고리즘, 웹검색, 생물정보학, 암호학