품질 정보를 이용한 서열 배치 알고리즘
(Sequence Alignment Algorithm using Quality Information)

나 중 채 † 노 강 호 † 박 근 수 ††
(Joong Chae Na) (Kangho Roh) (Kunsoo Park)

요 약 본 논문에서 다루는 문제는 품질 정보를 가지는 서열을 배치(alignment)하는 알고리즘이다. 시퀀싱(sequencing) 작업의 일부인 염기 결정 프로그램(base-calling program)에 의해서 생성되는 DNA 서열은 각 염기가 어느 정도 신뢰할 수 있는 가를 나타내는 품질 정보를 가진다. 그러나 지금까지 개발된 서열 배치 알고리즘들은 이러한 품질 정보를 고려하지 않았다. 본 논문에서는 품질 정보를 가지는 두 서열의 배치를 평가하는 기준을 제시한다. 이 평가 기준에 의한 최적의 서열 배치는 동적 프로그래밍(dynamic programming) 기법에 의해서 찾을 수 있다.

키워드 : 서열 배치 알고리즘, 동적 프로그래밍, 생물정보학, DNA 서열 비교

Abstract In this paper we consider the problem of sequence alignment with quality scores. DNA sequences produced by a base-calling program (as part of sequencing) have quality scores which represent the confidence level for individual bases. However, previous sequence alignment algorithms do not consider such quality scores. To solve sequence alignment with quality scores, we propose a measure of an alignment of two sequences with quality scores. We show that an optimal alignment in this measure can be found by dynamic programming.

Key words : Sequence alignment, dynamic programming, bioinformatics, DNA sequence comparison

1. 서 론

서열 배치(sequence alignment) 문제는 두 개의 서열에서 가장 유사한 배치를 찾는 문제이다. 서열 배치가 DNA나 단백질 서열 같은 생물학적인 데이타들의 유사성을 판단하는데 좋은 척도를 제공하기 때문에, 서열 배치 문제는 계산 분자 생물학(computational molecular biology) 분야에서 매우 중요하다[1-4]. 그래서 그동안 다양한 종류의 배치 개념과 알고리즘들이 개발되어왔다. 초기에는 두 서열 전체의 유사성을 측정하는 전역 배치(global alignment)에 대해서 연구가 진행되었고[5], 이후에 서열의 일부분에서의 유사성을 다루는 지역 배치(local alignment)와 셋 이상의 서열들 사이의 유사성을 다루는 다중 배치(multiple alignment)에 대해서 많은 연구가 진행되어 왔다[6-9]. 최근에도 앞서 서술한 기본적인 배치 개념들이 가지는 맹점을 보완하는 정규화 된 배치(normalized alignment) 등 다양한 연구들이 진행되고 있다[10-12].

본 논문에서는 품질 정보를 가지는 서열들의 배치 문제를 다룬다. 실제 사용되는 생물학적 데이타에는 많은 오류(error)들이 존재한다. 예를 들어, 어떤 생물들의 염기 서열을 밝혀내는 시퀀싱(sequencing)에서 사용되는 *PHRED*[13]같은 염기 결정(base-calling) 프로그램은 DNA 서열과 함께 각 염기(base)가 어느 정도 확실한지를 나타내는 품질 정보(quality information)를 생성한다. 즉, 품질 정보는 생성된 각 염기가 실제 DNA 서열의 염기와 같을 확률을 나타낸다. 그러나 지금까지 개발된 서열 배치 알고리즘들은 이러한 품질 정보를 전혀 고려하지 않는다. 즉, 기존의 알고리즘은 각 염기가 실제 DNA 서열의 염기가 확실하다는 가정 아래 가장 유사한 배치를 찾는다. 그래서 기존의 알고리즘은 품질 정보를 가지는 서열들의 배치를 찾는 목적에 적당하지 못하다. 즉, 기존의 알고리즘이 찾은 가장 유사한 배치는 품질 정보가 나타내는 확률을 고려할 경우 더 이상 가장 유사한 배치가 아닐 수도 있다.

품질 정보를 고려하지 않는 경우의 배치의 유사도와

† 비 회 원 : 서울대학교 전기.컴퓨터공학부
jcna@theory.snu.ac.kr
khroh@theory.snu.ac.kr
†† 종신회원 : 서울대학교 전기.컴퓨터공학부 교수
kpark@theory.snu.ac.kr
논문접수 : 2004년 6월 7일
심사완료 : 2005년 8월 10일

품질 정보를 고려하는 경우의 배치의 유사도가 서로 다르기 때문에, 일부 응용 프로그램에서는 기존의 품질 정보를 고려하지 않는 알고리즘 의해 가장 유사하다고 판단되는 배치를 찾은 후, 품질 정보를 이용하여 유사도를 재평가한다. 이는 품질 정보를 가지는 서열을 다루는 *PHRAP*[14], *ARACHNE*[15,16] 같은 응용 프로그램에서 많이 사용되는 기법이다. 하지만, 이 방법은 기존 알고리즘이 찾은 배치가 품질 정보를 고려하였을 때 얼마나 유사한 배치인지를 판단할 수 있으나, 품질 정보를 고려한 가장 유사한 배치를 찾지 못한다.

본 논문에서는 품질 정보를 가지는 서열의 최적 배치를 찾는 알고리즘을 최초로 제시한다. 이를 위해 우선 확률론과 기존의 평가기준에 입각하여 품질 정보를 가지는 두 서열의 유사도를 측정하는 새로운 평가 기준을 제안하고, 이 기준에 의한 가장 유사한 배치를 찾는 알고리즘을 제시한다. 이 알고리즘은 기존의 알고리즘처럼 동적 프로그래밍 기법을 사용한다. 본 논문에서는 전역 배치(global alignment)에 대해서만 다루지만, 본 논문의 결과는 지역 배치와 같은 다른 종류의 배치들에도 쉽게 적용될 수 있다.

본 논문에서 고려하는 문제와 기존의 알고리즘들이 다루는 문제사이의 관계에 대해서 살펴보자. 기존의 알고리즘들이 다루는 문제는 본 논문에서 고려하는 문제의 특수한 경우이다. 기존의 문제를 본 논문에서 고려하는 문제로 해석해보면, 기존의 알고리즘들은 각 염기의 에러 확률이 0인 서열을 다루거나, 서열의 각 염기의 에러 확률이 0라는 가정 하에 문제의 해를 제시한다. 따라서 새로 제시되는 알고리즘은 기존의 알고리즘의 결과를 내포하고 있어야 한다. 본 논문에서는 기존의 평가기준에서 사용하는 유사도 점수의 기대값을 새로운 평가 기준으로 제시한다. 이 기준은 위의 조건을 충족시키면서 품질 정보가 가지는 확률의 의미를 잘 반영한다. 본 논문의 알고리즘과 기존의 알고리즘 사이의 이러한 관계는 실험을 통해서 확인할 수 있다.

본 논문의 구성은 다음과 같다. 2절에서 품질 정보를 고려하게 된 생물학적 배경지식과 품질 정보를 고려하지 않는 기존의 결과를 소개하고, 다음 절에서 본 논문의 결과인 품질 정보를 가지는 서열의 배치를 평가하는 기준과 이에 기반을 둔 서열 배치 알고리즘을 제시한다. 4절에서는 품질 정보를 가지는 실제 서열을 이용하여 실험한 결과를 제시하고 이 실험 결과로부터 본 논문의 결과가 기존의 결과와 어떤 관계가 있는지 조명해 본다.

2. 배경지식

본 절에서는 생물학 데이타의 품질 정보가 무엇을 의미하는 지와 이를 고려하게 된 생물학적 배경에 대해서 기술한다. 다음으로 품질 정보를 이용하지 않는 기존의 전역 배치(global alignment) 알고리즘을 소개하고, 품질 정보를 고려하지 않는 기존의 전역 배치를 품질 정보를 가지는 서열에 적용하였을 때 발생할 수 있는 문제점을 예를 통해 살펴본다.

2.1 품질 정보

시퀀서(sequencer)라 불리는 기계는 DNA 조각을 읽어서 *trace* 데이타를 생성한다. 이 trace데이타는 염기들의 품질 정보를 포함한다. 그림 1은 trace의 데이타의 예이다. Trace 데이타는 4개의 곡선으로 구성되는데, 각각의 곡선은 4개의 염기들 A, C, T, G에 대한 신호를 나타낸다. 가로축은 거리를 나타내고 세로축은 데이타에 대한 신뢰도를 나타낸다. 즉, 높은 정점(peak)을 가지는 데이타일수록 더 정확하다. 곡선 위쪽의 문자들은 trace 데이타로부터 추정되는 원본 염기 서열이다.

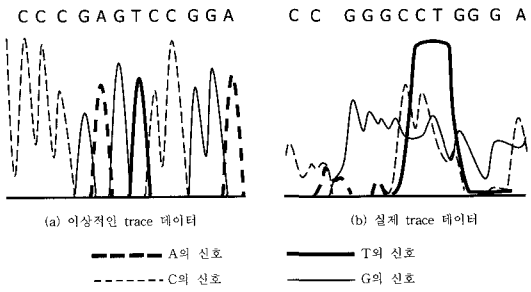


그림 1 trace 데이타의 예

이상적인 경우에는 trace 데이타에서 모든 정점이 동일한 거리를 두고 떨어져 있고, 서로 겹치지 않는다. 이러한 이상적인 데이타에서는 DNA 서열을 정확히 생성할 수 있다. 그림 1(a)는 이상적인 trace 데이타의 예이다. 정점들 사이의 거리들이 거의 동일하고, 어떤 위치에서 한 곡선의 정점이 다른 세 곡선의 정점보다 훨씬 높다. 그러나 실제 trace 데이타는 원시 데이타를 생성하는 생물학적인 실험 자체에 오류들이 존재하기 때문에 이상적인 것과 많이 다르다[13]. 첫째, 정점들 사이의 거리가 그림 1(b)에서처럼 매우 다양하다. 예를 들어 첫 번째에 위치한 C와 두 번째에 위치한 C 사이의 거리는 두 번째에 위치한 C와 세 번째에 위치한 G 사이의 거리보다 더 가깝다. 먼 거리는 그 사이에 어떤 염기가 누락되었을 수도 있다는 것을 시사한다. 둘째, 둘 이상의 곡선이 비슷한 정점을 가지고 있을 수 있다. 이런 경우 그 위치의 염기가 어떤 것인지 확실하지 않다. 셋째, 네 곡선의 정점이 모두 매우 낮은 경우가 있다. 이 경우도 우리는 그 위치의 염기가 무엇인지 확신할 수 없게 된다.

염기 결정 프로그램은 trace 데이타를 읽어서 염기들의 서열을 생성한다. 그리고 대부분의 염기 결정 프로그램은 생성하는 각 염기에 해당하는 품질 정보도 생성한다. 대부분의 염기 결정 프로그램이 비슷한 데이타를 생성하므로, 본 논문에서는 잘 알려진 염기 결정 프로그램 중 하나인 PHRED[13,14]를 기준으로 출력 데이타의 성질을 기술한다. 그림 1의 윗부분에 있는 문자들은 PHRED가 trace데이타를 읽어 들여서 생성한 염기 서열이다. PHRED는 염기들의 서열과 함께 품질 점수라 불리는 수들의 서열을 생성한다. PHRED의 품질 점수는 0~99 사이의 값을 가지는데, 그 위치의 염기가 실제와 다를 확률과 관계가 있다. 품질 점수를 Q라 하고 염기의 에러 확률을 P라 하면 $Q=-10\times\log_{10}P$ 인 관계가 성립한다. 예를 들어 PHRED가 생성한 서열에서 어떤 위치의 염기가 C이고 그에 해당하는 품질 점수를 20이라면, 그 위치의 염기가 C가 아닐 확률은 0.01이다. 염기 서열과 품질 점수를 생성한 후에 PHRED는 품질 점수를 이용해 서열을 정돈(trimming)하는 과정을 수행한다. 보통 trace 데이타의 처음 부분과 끝 부분은 생물학적 실험의 한계로 인해 앞에서 설명한 오류들을 더 많이 포함하고 있고, 결과적으로 품질 점수가 매우 낮다. 이렇게 많은 오류를 포함하는 양 끝 부분은 이 서열을 이용한 실험에서 좋지 않은 영향을 끼치기 때문에 잘라낸다. 그래서 실제 사용되는 서열에서 10보다 작은 품질 점수를 가지는 염기의 비율은 2~5% 정도이다. 다음은 PHRED에서 출력된 실제 DNA 서열과 품질 점수의 예이다. PHRED의 입력으로 시퀀서에서 생성된 실제 trace 데이타를 이용하였다.

A G C T T G C A T G C C T G C A G G T C G A C T C ...

13 16 19 22 16 20 16 22 29 39 39 39 34 34 29 29 29 33 29 35 33 35 35 45 45 ...

2.2 전역 배치

서열이란 알파벳 Σ에 속한 문자들의 나열이다. 본 논문에서는 DNA 서열에 대해 다루고 있으므로 Σ를 $\{a,c,t,g\}$로 가정한다. 공백(space)은 $\Delta\notin\Sigma$로 표기한다. 서열 A의 i번째 문자는 A_i, 부분 서열 $A_i A_{i+1}\cdots A_j$는 $A[i..j]$로 표기한다.

길이가 각각 m과 n인 두 서열 $A=A_1 A_2\cdots A_m$과 $B=B_1 B_2\cdots B_n$이 주어졌을 때, 두 서열의 *전역 배치(global alignment)*는 $A^*=A_1^* A_2^*\cdots A_l^*$와 $B^*=B_1^* B_2^*\cdots B_l^*$ $(n,m\le l)$이다. A^*와 B^*는 각각 A와 B의 문자들 사이에 0개 이상의 공백(Δ)을 삽입함으로써 만들어진다. 문자 쌍 A_i^*와 B_i^*는 그 문자가 무엇인지에 따라서 다음 세 종류의 *매핑(mapping)* 중 하나로 분류된다.

- 일치(match): $A_i^*\neq\Delta$, $B_i^*\neq\Delta$이고, $A_i^*=B_i^*$인 경우.
- 불일치(mismatch): $A_i^*\neq\Delta$, $B_i^*\neq\Delta$이고, $A_i^*\neq B_i^*$인 경우
- 갭(gap): A_i^* 또는 B_i^*가 Δ인 경우

$A_i^*=B_i^*=\Delta$인 경우는 허용하지 않는다. 각 매핑은 해당되는 점수를 가지는데, 일치는 γ, 불일치는 δ, 갭은 μ를 가진다. 이 γ, δ, μ를 매핑 점수 인자(parameter)라 부르는데, 응용(application)에 따라 다양한 값을 가진다. 보통 γ는 양수이고, δ와 μ는 음수이다.

배치의 점수 $S(A^*,B^*)$는 배치를 이루는 모든 문자쌍들의 매핑 점수의 합으로 정의된다. 즉, i번째 문자쌍 A_i^*와 B_i^*의 점수를 $S(A_i^*,B_i^*)$로 표기하면,

$$S(A^*,B^*)=\sum_{i=1}^{l}S(A_i^*,B_i^*)$$

이다.

예제 1. $A=a\ a\ t$, $B=c\ a\ a\ t$라고 하자. 다음은 A와 B의 전역 배치 중 하나이다.

$$\begin{array}{c} A^*=a\,a\,\Delta\,t \\ \quad\ |\ |\ |\ | \\ B^*=c\,a\,a\,t \end{array}$$

위 배치에서 첫 번째는 불일치 매핑이고, 두 번째와 네 번째는 일치 매핑이고, 세 번째는 갭 매핑이다. 매핑 점수 인자를 $\gamma=1$, $\delta=-1$, $\mu=-2$라고 가정하면, 이 배치의 점수는 $-1+1-2+1=-1$이다. □

알고리즘 1 서열 배치 문제를 풀기 위한 점화식(recurrence)

$\gamma>0$: 일치 점수

$\delta<0$: 불일치 점수

$\mu<0$: 갭 점수

$$S(A_i,B_j)=\begin{cases}\gamma & \text{if } A_i\text{와 } B_j\text{가 일치}\\ \delta & \text{if } A_i\text{와 } B_j\text{가 불일치}\end{cases}$$

$$H_{0,0}=0$$

$$H_{i,0}=H_{i-1,0}+\mu,\ (1\le i\le m)$$

$$H_{0,j}=H_{0,j-1}+\mu,\ (1\le j\le n)$$

$$H_{i,j}=\max\begin{cases}\mathrm{H_{i-1,j-1}+S(A_i,B_j)}\\ \mathrm{H_{i-1,j}}+\mu\\ \mathrm{H_{i,j-1}}+\mu\end{cases}$$

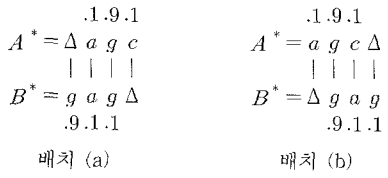
$$(1\le i\le m,\ 1\le j\le n)$$

그림 2 서열 배치 문제를 풀기 위한 점화식

서열 A, B와 매핑 점수 인자 γ, δ, μ가 입력으로 주어졌을 때, 서열 배치 문제(sequence alignment

problem)는 A와 B의 모든 가능한 배치 중에서 배치 점수가 가장 큰 배치를 찾는 것이다. 점수가 가장 큰 배치를 *최적 배치*(optimal alignment)라 한다. 이 문제는 동적 프로그래밍(dynamic programming) 기법을 이용하여 풀 수 있다. 이 기법은 크기가 $(m+1)\times(n+1)$인 테이블(table)을 사용하여 $O(mn)$ 시간에 A와 B의 최적 배치를 구한다. $H_{i,j}$를 $A[1..i]$와 $B[1..j]$의 최적 배치의 점수라 할 때, $H_{i,j}$는 알고리즘 1의 점화식을 이용하여 계산할 수 있고, 최적 배치는 테이블의 정보를 역으로 추적하여 찾을 수 있다. 알고리즘에 대한 좀더 자세한 설명은 [5]를 참조하기 바란다.

다음으로 위의 알고리즘을 품질 정보를 가지는 서열에 적용하였을 경우에 발생할 수 있는 문제점을 예를 통해 살펴본다. 아래 그림 3은 두 서열 $A=a\ g\ c$와 $B=g\ a\ g$의 배치들을 나타낸다. 각 염기의 위와 아래에 표기된 수는 염기가 맞을 확률을 나타낸다. 실제로 0.1 같이 매우 작은 확률을 가지는 경우는 매우 드물지만, 품질 정보를 고려하였을 때와 고려하지 않았을 때의 차이를 쉽게 알아 볼 수 있도록 작은 확률을 가지는 서열을 예로 든다. 배치 (a)는 2개의 일치와 2개의 갭으로 이루어져 있고, 배치 (b)는 1개의 일치, 1개의 불일치, 2개의 갭으로 이루어져 있다. 단순히 매핑의 종류만을 고려하면 분명 배치 (a)가 더 좋은 배치이다. 하지만 각 배치가 나타날 확률을 계산해보면, 배치 (a)에서 일치는 작은 확률로 나타나는 반면, 갭이 나타날 확률은 매우 크다. 배치 (b)에서는 반대로 일치가 나타날 확률은 크고, 갭이 나타날 확률은 매우 작다. 즉, 실제 각 염기가 맞을 확률을 고려할 경우에는 배치 (b)가 더 좋은 배치이다.



배치 (a) 배치 (b)

그림 3 서열 A=a g c와 B=g a g의 배치

3. 품질 정보를 이용한 서열 배치

이 절에서는 품질 정보를 이용한 서열 배치에 대해서 기술한다. 우선 품질 정보를 가지는 서열의 정의와 성질에 대해서 기술한 후, 품질 정보를 가지는 서열의 배치와 배치 점수를 어떻게 정의할 수 있는지 살펴본다.

품질 정보를 가지는 서열 $A=A_1A_2\cdots A_n$은 각 A_i가 Σ의 문자 중 하나이고 그 문자의 품질 점수가 Q_{A_i}인 서열이다. 품질 점수 Q_{A_i}는 A_i의 에러 확률이 $10^{-Q_{A_i}/10}$이라는 것을 의미한다. 앞으로 품질 정보를 가지는 서열을 품질서열이라 간략히 부르고, 이와 구별하여 품질 정보를 가지지 않는 서열을 일반서열이라 칭한다.

예제 2. 품질서열 $A=a\ t\ g$라 하고, 각 문자의 품질 점수를 차례대로 10, 20, 30이라 하자. 이때 각 문자의 에러 확률은 다음과 같다.

서열	a	t	g
품질 점수	10	20	30
에러 확률	0.1	0.01	0.001

□

문자의 품질 점수의 의미를 좀더 자세히 살펴본다. 서열에서 어떤 위치의 문자가 $x\in\Sigma$이고, 그 문자의 품질 점수가 Q_x라고 하자. 이는 그 위치에는 $1-10^{-Q_x/10}$의 확률로 x가 나타나고, $10^{-Q_x/10}$의 확률로 다른 문자가 나타나거나 혹은 공백으로 남는다는 것을 의미한다. 여기서 품질서열의 문자가 없어져서 생기는 공백을 2.2절의 배치에 삽입되는 공백(Δ)과 구별하기 위해서 '-'로 표시한다. 앞으로 x를 그 위치에서의 *대표문자*(*representative character*)라 칭한다. 본 논문에서는 다음을 가정한다.

가정 1: 어떤 위치에 대표문자가 나타날 확률은 보통 0.9보다 크다. 2절에서 설명했듯이 실제 사용되는 품질서열의 품질 점수는 10보다 크고, 이는 대표문자가 나타날 확률이 0.9보다 크다는 것을 의미한다.

가정 2: 대표문자 이외의 문자들과 공백(-)이 나타날 확률은 모두 같다. 예를 들어, 어떤 위치에 대표문자가 a이고, 에러 확률이 0.1이라면, 그 위치에 c가 나타날 확률, t가 나타날 확률, g가 나타날 확률, 공백(-)으로 남을 확률이 모두 0.025이다. 대표문자가 나타날 확률이 매우 크므로 다른 문자들이 나타날 확률은 매우 작다. 따라서 이 확률들을 모두 같다고 가정하더라도 실제 각 문자들이 나타날 확률과 크게 차이가 나지 않는다.

예제 3. $A=a\ t\ g$라 하고, 각 문자의 품질 점수를 차례대로 10, 20, 30이라 하자. 이때 각 위치에 문자들이 나타날 확률은 다음 표와 같다.

서열 위치	1	2	3
대표문자	a	t	g
품질 점수	10	20	30
a가 나타날 확률	0.9	0.0025	0.00025
c가 나타날 확률	0.025	0.0025	0.00025
t가 나타날 확률	0.025	0.99	0.00025
g가 나타날 확률	0.025	0.0025	0.999
공백(-)일 확률	0.025	0.0025	0.00025

□

품질서열의 전역 배치에 대해서 설명한다. 길이가 각각 m과 n인 두 품질서열 $A=A_1A_2\cdots A_m$과 $B=B_1B_2\cdots B_n$이 주어졌을 때, 두 품질서열의 전역 배치는 $A^*=A_1^*A_2^*\cdots A_l^*$와 $B^*=B_1^*B_2^*\cdots B_l^*$ $(n,m\le l)$이다. A^*와 B^*는 각각 A와 B의 문자들 사이에 0개 이상의 공백(Δ)을 삽입함으로써 만들어진다. 배치의 정의나, 공백(Δ)을 삽입해서 배치가 만들어진다는 것은 일반서열과 같다. 배치에 삽입되는 공백(Δ)은 그 부분에 항상 어떤 문자도 나타나지 않는다는 것을 의미하므로, 그 위치에 문자 $x\in\Sigma$가 나타날 확률은 0이고 공백(-)이 나타날 확률은 1이다. 문자 쌍 A_i^*와 B_i^*는 그 대표문자가 무엇인지에 따라서 다음 세 종류의 매핑 중 하나로 분류된다.

- 정규일치(regular-match): $A_i^*\neq\Delta$, $B_i^*\neq\Delta$이고, A_i^*와 B_i^*의 대표문자가 같은 경우
- 정규불일치(regular-mismatch): $A_i^*\neq\Delta$, $B_i^*\neq\Delta$이고, A_i^*와 B_i^*의 대표문자가 다른 경우
- 정규갭(regular-gap): A_i^* 또는 B_i^*가 Δ인 경우

$A_i^*=B_i^*=\Delta$인 경우는 허용하지 않는다. 위 세 매핑을 *품질매핑*이라 칭하고, 일반서열의 매핑인 일치, 불일치, 갭을 *일반매핑*이라 칭한다.

문자 쌍 A_i^*와 B_i^*의 매핑 점수 $S(A_i^*,B_i^*)$는 일반매핑 점수의 기대값으로 정의한다. A_i^*와 B_i^*의 실제 문자가 무엇이냐에 따라 품질매핑은 일반매핑인 일치, 불일치, 갭 중 하나가 된다. 다음 표는 A_i^*와 B_i^*의 실제 문자에 따라 어떤 일반매핑이 되는지를 나타낸다.

B_i^* \ A_i^*	a	c	t	g	-
a	M	N	N	N	G
c	N	M	N	N	G
t	N	N	M	N	G
g	N	N	N	M	G
-	G	G	G	G	E

- M: 실제 문자가 같은 경우이다. 따라서 이 경우는 일치 점수 γ를 가진다.
- N: 둘 다 공백이 아니면서 실제 문자가 서로 다른 경우이다. 따라서 불일치 점수 δ를 가진다.
- G: 한쪽은 일반 문자이고 다른 한쪽은 공백(-)인 경우이다. 이 경우는 일반 매핑에서 갭과 발생한 원인은 다르지만, 같은 의미를 가지므로 갭 매핑으로 간주하고 점수 μ를 부여한다. 예를 들어, A_i^*의 실제 문자로 a가 나타나고, B_i^*에는 어떤 문자도 나타나지 않았다고 하자. 이는 일반서열에서 갭 매핑에 해당하고, 배치의 점수를 계산할 때 이 매핑의 점수는 μ가 된다.
- E: A_i^*와 B_i^*가 모두 공백인 경우이다. 이 경우엔 이 매핑이 배치 상에서 없는 것으로 간주할 수 있으므로 배치 점수에 영향을 주지 않도록 0의 점수를 부여한다.

따라서 일치 매핑이 될 확률을 $P_m(A_i^*,B_i^*)$, 불일치 매핑이 될 확률을 $P_n(A_i^*,B_i^*)$, 갭 매핑이 될 확률을 $P_g(A_i^*,B_i^*)$라 하면,

$$S(A_i^*,B_i^*)=\gamma\times P_m(A_i^*,B_i^*)+\delta\times P_n(A_i^*,B_i^*)+\mu\times P_g(A_i^*,B_i^*)$$

이다.

각 품질매핑의 점수를 구체적으로 구해보자. A_i^*의 실제 문자가 $x\in\Sigma$일 확률을 α_x, 공백으로 남을 확률을 α_-, B_i^*의 실제 문자가 $x\in\Sigma$일 확률을 β_x, 공백으로 남을 확률을 β_-로 표기한다.

- 정규일치(regular-match)의 경우

A_i^*와 B_i^*의 대표문자를 a라 하자. 가정 2로부터

$$\alpha_c=\alpha_t=\alpha_g=\alpha_-=\frac{1-\alpha_a}{4},$$

$$\beta_c=\beta_t=\beta_g=\beta_-=\frac{1-\beta_a}{4}$$

이다.

	a: α_a	c: $\frac{1-\alpha_a}{4}$	t: $\frac{1-\alpha_a}{4}$	g: $\frac{1-\alpha_a}{4}$	-: $\frac{1-\alpha_a}{4}$
a: β_a	$\alpha_a\beta_a$	X	X	X	X
c: $(1-\beta_a)/4$	Y	Z	Z	Z	Z
t: $(1-\beta_a)/4$	Y	Z	Z	Z	Z
g: $(1-\beta_a)/4$	Y	Z	Z	Z	Z
-: $(1-\beta_a)/4$	Y	Z	Z	Z	Z

* $X=\frac{(1-\alpha_a)\beta_a}{4}$, $Y=\frac{\alpha_a(1-\beta_a)}{4}$, $Z=\frac{(1-\alpha_a)(1-\beta_a)}{16}$

위 확률 표로부터 다음 식을 얻는다. Z는 매우 작은 값이기 때문에 아래 식에서 무시한다.

1. $P_m(A_i^*,B_i^*)=\alpha_a\beta_a+3Z\approx\alpha_a\beta_a$
2. $P_n(A_i^*,B_i^*)=3X+3Y+6Z\approx\frac{\alpha_a+\beta_a-2\alpha_a\beta_a}{4}\times3$
3. $P_g(A_i^*,B_i^*)=X+Y+6Z\approx\frac{\alpha_a+\beta_a-2\alpha_a\beta_a}{4}$

그러므로 정규일치의 매핑 점수는

$$S(A_i^*,B_i^*)=\gamma\times\alpha_a\beta_a+\delta\times\frac{3(\alpha_a+\beta_a-2\alpha_a\beta_a)}{4}$$

$$+\mu\times\frac{\alpha_a+\beta_a-2\alpha_a\beta_a}{4}$$

이다.

• 정규불일치(regular-mismatch)의 경우

A_i^*의 대표문자를 c, B_i^*의 대표문자를 a라 하자. 가정 2로부터

$$\alpha_a=\alpha_t=\alpha_g=\alpha_-=\frac{1-\alpha_c}{4},$$

$$\beta_c=\beta_t=\beta_g=\beta_-=\frac{1-\beta_a}{4}$$

이다.

	a: $\frac{1-\alpha_c}{4}$	c: α_c	t: $\frac{1-\alpha_c}{4}$	g: $\frac{1-\alpha_c}{4}$	-: $\frac{1-\alpha_c}{4}$
a: β_a	X	$\alpha_c\beta_a$	X	X	X
c: $(1-\beta_a)/4$	Z	Y	Z	Z	Z
t: $(1-\beta_a)/4$	Z	Y	Z	Z	Z
g: $(1-\beta_a)/4$	Z	Y	Z	Z	Z
-: $(1-\beta_a)/4$	Z	Y	Z	Z	Z

1) * $X=\frac{(1-\alpha_c)\beta_a}{4}$, $Y=\frac{\alpha_c(1-\beta_a)}{4}$, $Z=\frac{(1-\alpha_c)(1-\beta_a)}{16}$

위 확률 표로부터 다음 식을 얻는다. Z는 매우 작은 값이기 때문에 아래 식에서 무시한다.

1. $P_m(A_i^*,B_i^*)=X+Y+2Z\approx\frac{\alpha_c+\beta_a-2\alpha_c\beta_a}{4}$
2. $P_n(A_i^*,B_i^*)=\alpha_c\beta_a+2X+2Y+7Z\approx\frac{\alpha_c+\beta_a}{2}$
3. $P_g(A_i^*,B_i^*)=X+Y+6Z\approx\frac{\alpha_c+\beta_a-2\alpha_c\beta_a}{4}$

그러므로 정규불일치의 매핑 점수는

$$S(A_i^*,B_i^*)=\gamma\times\frac{\alpha_c+\beta_a-2\alpha_c\beta_a}{4}+\delta\times\frac{\alpha_c+\beta_a}{2}+\mu\times\frac{\alpha_c+\beta_a-2\alpha_c\beta_a}{4}$$

이다.

• 정규갭(regular-gap)의 경우

A_i^*의 대표문자를 a이고, B_i^*는 공백(Δ)이라 하자. B_i^*가 공백(Δ)이므로 $\beta_-=1$이다. 위와 같이 계산하면 정규갭의 매핑 점수는

$$S(A_i^*,B_i^*)=\mu\times(1-\alpha_-)=\mu\times\frac{3+\alpha_a}{4}$$

이다.

정규일치와 정규불일치의 매핑 점수 식을 좀더 단순화 시켜보자. 가정 1에 의해서 α_a와 β_a가 0.9 이상이므로, $\frac{\alpha_a+\beta_a-2\alpha_a\beta_a}{4}$는 항상 0.045 이하로 매우 작은 값이다. 따라서 이 항(term)을 정규일치의 식에서 생략하면, $S(A_i^*,B_i^*)=\gamma\times\alpha_a\beta_a$이다. 정규불일치의 경우도 비슷하게 값이 작은 항을 무시하면 $S(A_i^*,B_i^*)=\delta\times(\alpha_c+\beta_a)/2$ 이다.

일반서열에서처럼 품질서열의 배치의 점수 $S(A^*,B^*)$는 배치를 이루는 모든 문자 쌍들의 매핑 점수의 합으로 정의된다. 즉,

$$S(A^*,B^*)=\sum_{i=1}^{l}S(A_i^*,B_i^*)$$

이다.

예제 4. $A=a\ a\ t$, $B=c\ a\ a\ t$, A의 각 문자의 품질 점수를 차례로 20, 10, 10, B의 각 문자의 품질 점수를 차례로 10, 10, 20, 10이라 하자. 다음은 A와 B의 전역 배치중 하나이다.

$$\begin{array}{c}A^*=a\,a\,\Delta\,t\\ \quad\ |\ |\ |\ |\\ B^*=c\,a\,a\,t\end{array}$$

매핑 점수 인자를 $\gamma=1$, $\delta=-1$, $\mu=-2$라 가정하면, 위 배치에서 첫 번째는 정규불일치 매핑이고, 두 번째와 네 번째는 정규일치 매핑이고, 세 번째는 정규갭 매핑이다. 각 매핑의 점수는

$S(A_1^*,B_1^*)=-1\times(0.99+0.9)/2=-0.945$

$S(A_2^*,B_2^*)=1\times(0.9\times0.9)=0.81$

$S(A_3^*,B_3^*)=-2\times(3+0.99)/4=-1.995$

$S(A_4^*,B_4^*)=1\times(0.9\times0.9)=0.81$

이고, 따라서 이 배치의 점수는 $-0.945+0.81-1.195+0.81=-1.32$이다. □

알고리즘 2는 길이가 각각 m과 n인 두 서열이 주어졌을 때 배치 점수가 가장 큰 배치, 즉 최적 배치를 찾는다. $H_{i,j}$를 $A[1..i]$와 $B[1..j]$의 최적 배치의 점수라 할 때, $H_{i,j}$는 알고리즘 2의 점화식을 이용하여 계산할 수 있고, 최적 배치는 테이블의 정보를 역으로 추적하여 찾을 수 있다. 이 알고리즘은 앞에서 제시한 매핑 점수를 기준으로 최적 배치를 찾는 점화식으로 기존의 일반 서열의 최적 배치를 찾는 것과 같은 구조를 가지고 있다. 따라서 이 알고리즘은 $O(mn)$의 메모리와 $O(mn)$의 시간을 사용한다.

4. 실험 및 분석

이 절에서는 품질서열에 대해서 기존의 알고리즘과 본 논문에서 제시하는 알고리즘이 실제로 서로 다른 최적 배치를 찾는 경우가 얼마나 발생하는지, 그런 경우 어떤 알고리즘이 더 좋은 배치를 찾아내는지, 두 알고리즘이 어떤 차이를 보이는지를 알아보기 위해, 실제 데이타를 이용하여 실험한 결과를 제시하고, 실험 결과가 나

알고리즘 2 품질서열의 배치 문제를 풀기 위한 점화식 (recurrence)

$\gamma > 0$: 일치 점수

$\delta < 0$: 불일치 점수

$\mu < 0$: 갭 점수

Q_x : 문자 x의 품질 점수

$$P_x = 1 - 10^{-Q_x/10}$$

$$S(A_i, B_j) = \begin{cases} \gamma \times P_{A_i} P_{B_j} & \text{if } A_i\text{와 } B_j\text{가 정규일치} \\ \delta \times (P_{A_i} + P_{B_j})/2 & \text{if } A_i\text{와 } B_j\text{가 정규불일치} \end{cases}$$

$$H_{0,0} = 0$$

$$H_{i,0} = H_{i-1,0} + \mu \times \frac{3 + P_{A_i}}{4}, \ (1 \le i \le m)$$

$$H_{0,j} = H_{0,j-1} + \mu \times \frac{3 + P_{B_j}}{4}, \ (1 \le j \le n)$$

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S(A_i, B_j) \\ H_{i-1,j} + \mu \times \dfrac{3 + P_{A_i}}{4} \\ H_{i,j-1} + \mu \times \dfrac{3 + P_{B_j}}{4} \end{cases}$$
$$(1 \le i \le m, 1 \le j \le n)$$

그림 4 품질서열의 배치 문제를 풀기 위한 점화식

타내는 의미를 살펴본다.

실험은 다음과 같은 과정으로 진행되었다. 우선 NCBI 웹사이트(http://www.ncbi.nlm.nih. gov/Traces/trace.fcgi)에서 4개의 실제 유전체(genome) 데이타들을 받아왔다. 이 유전체 데이타의 이름은 "gnl|ti|14264831" (Data 1), "gnl|ti|222680126" (Data 2), "gnl|ti|312912383" (Data 3), "gnl|ti|125121088" (Data 4)이다. 각 유전체 데이타는 품질 점수를 가지는 20000개의 서열들로 구성되어 있다. 이 서열들의 길이는 1400~2000이다. 각 유전체 데이타의 서열들의 평균 품질 점수는 11.9(Data 1), 17.7(Data 2), 16.3(Data 3), 18.0(Data 4)이다.

각 유전체 데이타에서 2000개의 서열 쌍을 임의로 선택해 알고리즘 1과 알고리즘 2를 수행한 후, 결과를 비교하였다. 놀랍게도, 8000개의 서열 쌍 모두에 대해서 두 알고리즘의 결과가 서로 달랐다. 또 다른 실험으로 짧은 길이의 서열에 대한 결과를 알아보기 위해서 원래 서열을 작은 길이의 조각으로 자른 후에 같은 방법으로 실험을 수행하였다. 이 실험의 결과는 표 1에 제시되어 있다. 길이가 10인 서열 쌍에서 조차 두 알고리즘의 결과가 다른 경우가 대략 50% 정도였다.

두 알고리즘의 결과가 서로 다른 경우, 품질 정보가 나타내는 확률에 입각해 두 알고리즘에 의한 최적 배치를 비교 분석한 결과, 알고리즘 2가 더 유사한 배치를 찾는다는 것을 확인할 수 있었다. 이를 구체적으로 보여주는 두 개의 예를 제시한다. 이들은 짧은 서열들의 쌍에서 자주 나타나는 전형적인 예이다.

첫 번째 예는 그림 5에 나타나 있다. $A = a\ g\ c\ g\ c$, $B = t\ c\ t\ c\ t\ g$라 하고, 매핑 점수 인자를 $\gamma = 1$, $\delta = -1$, $\mu = -1$이라 했을 때, (a)와 (b)는 각각 알고리즘 1과 알고리즘 2에 의해서 계산된 최적 배치이다. 각 문자의 품질 점수는 A^*는 위에, B^*는 아래에 표기하였다. (a)와 (b)의 배치 모두 2개의 정규일치, 2개의 정규불일치, 3개의 정규갭으로 이루어져 있다. 우선, 정규일치를 이루고 있는 문자의 품질 점수를 살펴보면, (a)에서는 (11, 15)와 (11, 10)이고, (b)에서는 (11, 19)와 (11, 15)이다. 이는 (b)에서 구한 매핑이 일치가 될 확률이 더 높다는 것을 의미한다. 다음으로 정규불일치를 이루는 문자 쌍의 품질 점수를 살펴보면, (a)에서는 (10, 19)와 (12, 19)이고, (b)에서는 (10, 16)과 (11, 19)이다. 품질 점수를 이용해 계산해보면 (a)의 불일치 매핑이 (b)의 불일치 매핑보다 더 높은 확률로 나타난다는 것을 알 수 있다. 정규갭을 이루는 문자 쌍도 마찬가지이다.

(a) 알고리즘 1에 의한 최적 배치

```
          12 10 11   11 11
A* = Δ  a  g  c  Δ  g  c
     |  |  |  |  |  |  |
B* = t  c  t  c  t  g  Δ
     16 19 19 15 12 10
```

(b) 알고리즘 2에 의한 최적 배치

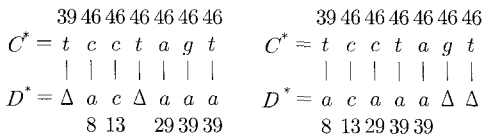
```
          12 10 11 11 11
A* = a  g  c  g  c  Δ  Δ
     |  |  |  |  |  |  |
B* = Δ  t  c  t  c  t  g
     16 19 19 15 12 10
```

그림 5 두 알고리즘의 결과가 서로 다른 예 1

표 1 짧은 서열에 대한 실험 결과

서열 길이	10		20		30	
두 알고리즘의 결과	같음	다름	같음	다름	같음	다름
Data 1	803	1197	205	1795	81	1919
Data 2	1319	681	629	1371	285	1715
Data 3	805	1195	219	1781	82	1918
Data 4	1143	857	497	1503	143	1857

이상을 종합해보면 (b)의 배치가 품질 정보를 고려하였을 때, 더 좋은 배치이다.



(a) 알고리즘 1에 의한 최적 배치 (b) 알고리즘 2에 의한 최적 배치

그림 6 두 알고리즘의 결과가 서로 다른 예 2

두 번째 예는 그림 6에 나타나 있다. 이 예에서도 매핑 점수 인자로 $\gamma=1$, $\delta=-1$, $\mu=-1$을 사용하였다. 두 배치 모두 2개의 일치, 3개의 불일치, 2개의 갭으로 이루어져 있다. 일치를 이루는 문자의 품질 점수를 살펴보면, (a)에서는 (46, 13)과 (46, 29)이고, (b)에서는 (46, 13)과 (46, 39)이다. 불일치를 이루는 문자의 품질 점수를 살펴보면, (a)에서는 (46, 8), (46, 39), (46, 39)이고, (b)에서는 (39, 8), (46, 29), (46, 39)이다. 따라서 일치와 불일치 매핑의 경우 (b)의 배치가 더 좋은 배치이다. 갭의 매핑을 이루는 품질 점수를 살펴보면, (a)는 39와 46이고, (b)는 46과 46이다. 갭의 경우 (a)의 배치가 (b)의 배치보다 더 좋지만, 이는 일치와 불일치 매핑에서 (b)의 우월성보다는 약하다. 그래서 전체 매핑을 고려해보면, (b)의 배치가 더 좋은 배치이다.

우리는 실험을 통해서 다음과 같은 사실을 알 수 있었다.

첫째, 실제 품질서열의 품질 점수가 높을수록, 즉 에러 확률이 낮을수록 알고리즘 1과 알고리즘 2에 의한 최적 배치가 같은 경우가 많았다. 이는 에러 확률이 낮을수록 기존의 알고리즘에서 가정하는 에러 확률 0에 근접하기 때문이다. 본 논문의 알고리즘은 기존의 알고리즘을 일반화한 것이다. 본 논문의 알고리즘의 점화식에서 에러 확률을 0으로 가정하면 기존의 알고리즘과 같은 점화식이 나오는데, 이를 뒤집어 생각하면 기존의 알고리즘은 본 논문에서 다루는 문제를 근사적으로 해결한다고 볼 수 있다. 이 경우는 근사 해가 최적 배치와 일치하는 경우이다.

둘째, 표 1에서 볼 수 있듯이 매우 짧은 서열에서 조차 알고리즘 1과 알고리즘 2에 의한 최적 배치가 서로 다른 경우가 많이 나타났다. 즉, 2절에서 지적했던 품질서열에 기존의 알고리즘을 적용했을 때 발생할 수 있는 문제가 실제 품질서열의 배치에서 많이 나타나고, 이는 품질서열의 에러 확률이 매우 작더라도 기존의 알고리즘이 확률을 고려했을 때의 최적 배치를 찾지 못하는 경우가 실제로 많이 발생한다는 것을 의미한다.

셋째, 알고리즘 1과 알고리즘 2에 의한 최적 배치가 서로 다른 경우, 기존의 알고리즘에 의해 평가해보면 두 배치가 같은 배치 점수를 가지는 경우가 대부분이었다. 그림 5, 6의 예도 이 경우에 해당한다. 즉, 각 예에서 배치 (a)나 배치 (b) 모두 알고리즘 1의 기준에서 보면 최적 배치 중 하나이다. 이는 위와 같은 맥락으로 생각해보면 그 이유를 쉽게 알 수 있다. 실제 서열에서 에러 확률이 대부분 0.1보다 작기 때문에 기존의 알고리즘이 본 논문에서 다루는 문제를 어느 정도는 근사적으로 해결한다고 볼 수 있다. 또 반대로 생각하면 이 결과는 본 논문에서 제시하는 일반화된 알고리즘이 기존의 알고리즘과 일맥상통하는 결과를 제시한다는 것을 의미한다.

5. 결 론

본 논문에서는 품질 점수를 가지는 서열의 배치를 구하는 알고리즘을 최초로 제시하였다. 품질 점수를 가지는 서열에서 품질 점수는 각 문자가 표시된 문자가 아닌 다른 문자일 확률을 나타내므로, 각 문자가 확실하다고 가정하는 일반적인 서열의 배치를 평가하는 기준과는 다른 기준이 필요하다. 본 논문에서는 일반서열의 점수 기준과 일관성(consistency)을 가지면서 품질서열의 배치를 적절하게(reasonable) 평가하는 기준을 처음으로 제시하였고, 이 새로운 기준에 의해서 가장 좋은 배치를 찾는 알고리즘을 제시하였다. 본 논문의 알고리즘의 시간과 공간 복잡도는 품질 정보를 고려하지 않는 기존의 알고리즘과 동일하다. 본 논문의 결과는 기존의 배치에 관련된 다양한 연구에 쉽게 접목될 수 있다는 장점을 가진다.

참 고 문 헌

[1] Waterman, M.S., Introduction to Computational Biology, Champman and Hall, 1995.

[2] Gusfield, D., Algorithms on Strings, Trees and Sequences: Computer science and Computational Biology, Cambridge University Press, 1997.

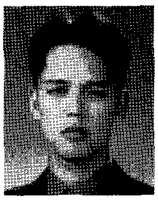
[3] Apostolico, A. and Giancarlo, R., "Sequence Alignment in Molecular Biology," Journal of Computational Biology 5(2), pp. 173-196, 1998.

[4] Pevzner, P., Computational Molecular Biology: An Algorithmic Approach, The MIT Press, 2000.

[5] Needleman, S.B. and Wunsch, C.D., "A general method applicable to the search for similarities in the amino acid sequences of two proteins," Journal of Molecular Biology 48, pp. 443-453, 1970.

[6] Gotoh, O., "An improved algorithm for matching biological sequences," Journal of Molecular Biology 162, pp. 705-504, 1982.

[7] Smith, T.F. and Waterman, M.S., Identification of Common Molecular Biology, PWS Publishing Company, 1997.
[8] Gusfield, D., "Efficient methods for multiple sequence alignment with guaranteed error bounds," Bulletin of Mathematical Biology 55, pp. 141-154, 1993.
[9] Hubbard, T., Lesk, A. and Tramontano, A., "Gathering them into the fold," Nature Structural Biology 4, pp 313, 1996.
[10] Zhang, Z., Berman, P., Wiehe, T. and Miller, W., "Post-processing long pairwise alignments," Bioinformatics 15(2), pp. 1012-1019, 1999.
[11] Arslan, A., Egecioglu, O. and Pevzner P., "A new approach to sequence comparison: Normalized sequence alignment," Bioinformatics 17(4), pp. 327-337, 2001.
[12] Crochemore M., Landau, G. and Ziv-Ukelson, M., "A sub-quadratic sequence alignment algorithm for unrestricted cost matrices," In 13th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 679-688, 2002.
[13] Ewing, B., Hillier, L., Wendl, M.C. and Green, P., "Base-calling of automated sequencer traces using phred. I. accuracy assessment," Genome Research 8(3), pp. 175-185, 1998.
[14] Green, P., Documentation for phrap, Genome Center, University of Washington, http://www.phrap.org/phrap.docs/phrap.html.
[15] Batzoglou, S., Jaffe, D., Stanley, K., Butler, J., Gnerre, S., Mauceli, E., Berger, B., Mesirov, J. and Lander E., "Arachne: A whole-genome shotgun assembler," Genome Research 12, pp. 177-189, 2002.
[16] Jaffe, D., Butler, J., Gnerre, S., Mauceli, E., Lindblan-Toh, K., Mesirov, J., Zody, M. and Lander E., "Whole-genome sequence assembly for mammalian genomes: Arachne 2," Genome Research 13, pp. 91-96, 2003.



나 중 채
1998년 서울대학교 컴퓨터공학과 학사
2000년 서울대학교 컴퓨터공학과 석사
2000년~현재 서울대학교 컴퓨터공학부 박사과정. 관심분야는 컴퓨터이론, 알고리즘, 생물정보학

노 강 호
2001년 서울대학교 컴퓨터공학과 학사
2001년~현재 서울대학교 컴퓨터공학부 석박통합과정 재학중. 관심분야는 컴퓨터이론, 알고리즘, 생물정보학

박 근 수
정보과학회논문지 : 시스템 및 이론 제 32 권 제 5 호 참조