# 효율적인 복수서열정렬 최적화기법†

김 °진 정우철 엄상용
한림대학교 정보통신공학부, 한림대학교 컴퓨터공학과
jinkim@hallym.ac.kr, wcjung@center.cie.hallym.ac.kr suhmn@inc.hallym.ac.kr

# An efficient optimization method for multiple sequence alignment

Jin °Kim Woocheol Jung Saangyong Uhmn
Divison of Information and Communication Engineering, Department of Computer Engineering

**요 약**

단백질들의 복수서열정렬은 단백질 서열간의 관계를 유추할 수 있는 유용한 도구이다. 최적화된 복수서열정렬을 얻기 위해 사용되는 가장 유용한 방법은 dynamic programming이다. 그러나 dynamic programming은 특정한 비용함수를 사용할 수 없기 때문에 특별한 경우 최적의 복수서열정렬을 제공하지 못하는 문제점이 있다. 우리는 이러한 문제점을 해결하기 위하여 부분정렬개선기법을 사용한 알고리즘을 제안하였으며, 이 알고리즘이 dynamic programming의 문제점을 효과적으로 해결함을 보였다.

## 1. 서 론

DNA, RNA 및 단백질들의 서열(sequence)들은 유전에 관한 중요한 정보를 가지고 있다. 서열 순서확인(sequencing)이란 유전 물질로부터 서열의 글자를 읽어내는 작업이라 정의할 수 있다. 서열 순서확인에 의해 얻어진 서열들을 서로 비교 정렬하는 방법을 서열정렬(sequence alignment)라 하는데, 서열정렬 방법은 서열들사이의 유사도를 계산해 준다. 두 개의 서열을 비교 정렬하는 방법을 쌍정렬(pairwise alignment)이라고 하며, 두 개 이상의 서열을 비교 정렬하는 방법을 복수서열정렬(multiple sequence alignment)이라고 한다[2-14]. 두 개의 서열을 정렬하는 최적의 방법은 Needleman과 Wunch[1]에 의해 소개된 dynamic programming 기법에 의해 해결되었다.

그러나 dynamic programming 방법은 특정 비용 함수를 이용하지 못하는 단점을 갖는다. 이 논문에서는 이러한 dynamic programming의 단점을 해결하기 위한 방법을 제시하였다. 2장에서는 복수서열정렬문제를 보다 공식적으로 정의하였고, 복수서열정렬에 사용되는 비용함수를 소개하고, dynamic programming의 문제점을 기술하였다. 3장에서는 이러한 문제점을 해결하기 위한 알고리즘을 기술하였다. 4장에서는 새로운 알고리즘에 의해 얻어진 실험결과들을 보였으며 5장에서 결론을 내렸다.

## 2. 복수서열정렬

### 2.1. 서열정렬을 위한 정의

복수서열 정렬문제를 위해, 다음과 같은 정의를 사용하기로 한다.

- 알파벳(alphabet) $\Sigma'$은 문자들과 널(null,'-')로 이루어진 유한 집합이다($\Sigma' = \Sigma \cup \{'-'\}$). DNA에서 사용되는 문자 집합은 $\Sigma$={a, t, g, c}, RNA인 경우 $\Sigma$={a, u, g, c}이며 단백질의 경우 $\Sigma$={W, Y, F, V, L, I, M, K, R, H, Q, E, D, N, G, A, P, T, S, C}이다. 널의 생물학적인 의미는 서열에 삽입이 발생했거나, 비교된 서열에서 삭제가 발생했음을 의미한다.
- 서열(sequence)은 $\Sigma$로 이루어진 유한한 길이의 스트링이다.
- 갭(gap)은 유한한 길이의 연속된 널이다.
- 입력서열 $S_1, S_2, \cdots, S_k$은 각각 길이 $n_1, n_2, \cdots, n_k$의 $k$개의 입력 서열들이며($S_1 = s_{11}s_{12}\cdots s_{1n_1}$, $S_2 = s_{21}s_{22}\cdots s_{2n_2}$, $\cdots$, $S_k = s_{k1}s_{k2}\cdots s_{kn_k}$), 알파벳은 $\Sigma$이며, $s_{ij} \in \Sigma$는 $i$번째 서열의 $j$번째 원소를 나타낸다($1 \le i \le k$, $1 \le j \le n_i$).

서열정렬은 $\Sigma'$으로 이루어진 동일한 길이 $l$의 의사 서열 $S'_1 = s'_{11}s'_{12}\cdots s'_{1l}$, $S'_2 = s'_{21}s'_{22}\cdots s'_{2l}$, $\cdots$ , $S'_k = s'_{k1}s'_{k2}\cdots s'_{kl}$을 얻는 것이다. 이 때 널이 포함된 의사 서열 $S'_i$로부터 널을 제거하면 원래의 서열 $S_i$를 얻을 수 있다. 얻을 수 있는 서열정렬의 개수는 Kim[14]의 논문에 자세히 설명되어 있다.

### 2.2. MSA

MSA는 Lipman 등에 의해 1989년 작성 발표된 프로그램으로 복수서열정렬을 얻기 위한 최적화 알고리즘 가운데 가장 표준적인 알고리즘인 dynamic programming 알고리즘을 이용한다[15]. MSA는 최적 혹은 최적에 가까운 점수를 가진 복수서열정렬을 제공하나 서열의 평균 길이가 $l$이고 서열의 개수가 $n$일때 $O((2n)^l)$의 시간복잡도에서 알 수 있듯이 정렬하려는 서열의 개수가 증가함에 따라 실행시간이 지수함수적으로 증가하는 문제점을 갖는다. 따라서 이 시간증가의 문제점을 해결하기 위한 여러 방법이 제안되었다.(경험적인 서열정렬 알고리즘, quasi-natural gap cost 비용함수)

### 2.3. 비용함수

복수서열정렬에 있어서 작은 비용을 가지는 복수서열정렬이 큰 비용을 가지는 복수서열정렬보다 생물학적현상을 잘 표현할 수 있는 적절한 비용함수를 이용하여 한다. Altschul[16]은 복수서열정렬을 위한 몇 가지 비용함수를 분류하였는데, 이들 비용함수는 교체비용(substitution cost)와 갭비용(gap cost)으로 구성된다. 즉, 복수서열정렬 $A$의 비용은 다음 식에서 보듯이 교체비용과 갭 비용의 합으로 계산된다.

$$\text{비용}(A) = \text{교체비용}(A) + \text{갭의개수}(A) \cdot gap\ penalty$$

#### 2.3.1. 교체비용

이 논문에서 사용된 교체 비용은 SP(Sum of Pairs) 교체비

† 본 연구는 정보통신부 정보통신선도기반기술개발사업의 지원에 의하여 이루어진것임.

용이다. SP 교체비용은 $n$개의 서열이 정렬되었을 때, $n(n-1)/2$개의 쌍에 대한 교체비용의 합이다. 즉, 복수서열정렬 $A$의 교체비용은 다음의 식에 의해 계산될 수 있다.

$$\text{교체비용}(A) = \sum_{i=1}^{n-1}\sum_{j=i}^{n} \text{교체비용}(S'_i, S'_j) \cdot weight(i,j)$$

이때 $weight(i,j)$는 서열 $S'_i$와 $S'_j$를 비교하여 얻은 교체비용에 대한 가중치이며, 단백질 서열간의 교체비용으로는 Dayhoff matrix가 주로 이용된다.[17].

### 2.3.2. 갭 비용

갭 비용은 연속된 널('-')의 개수에 부과되는 비용이다. Altschul은 생물학적으로 자연스러운 갭 비용함수로써 natural gap cost를 제안하였다[16]. MSA에서 서열정렬비용 계산에 natural gap cost를 사용하기 위해서는 전 상태의 정렬기록(alignment history)를 가지고 있어야 한다. $n$개의 서열을 SP 교체비용과 natural gap cost를 사용하여 정렬할 경우, MSA가 보유하고 있어야 하는 기록(history)의 양, $H_A(n)$은 다음의 순환식으로 표현될 수 있다[16].

$$H_A(n) = \begin{cases} 1 & n = 0 \\ \sum_{i=0}^{n-1}\binom{n}{i} H_A(n-1) & n \geq 1 \end{cases}$$

위의 수식에 따라 10개의 서열의 기록 보유 공간을 계산하면 다음의 표와 같다.

| gap cost / 서열 개수 | natural gap cost | quasi-natural gap cost |
|---|---|---|
| 10 | 102,247,563 | 1023 |

Altschul은 공간문제를 해결하기 위하여 quasi-natural gap cost를 제안하였다[16].

이 논문에서 MSA에 의해 제공되는 정렬이 가진 점수를 $Opt_{quasi\text{-}natural}$로 정의한다. 또한 natural gap cost를 적용하여 얻은 최적의 점수를 $Opt_{natural}$로 정의한다.

아래의 표 1은 natural gap cost와 quasi-natural gap cost를 사용했을 때의 계산된 갭의 개수를 보여주고 있다.

| gap cost / 서열 | X---X<br>XXX-X<br>XXXXX | X---X<br>XX-XX<br>XXXXX |
|---|---|---|
| natural gap cost | 3 | 3 |
| quasi-natural gap cost | 3 | 4 |

표 1 natural gap cost와 quasi-natural gap cost의 차이

위 표 1의 예에서 볼 수 있듯이 한 서열에 포함된 갭이 다른 서열의 갭에 완전히 포함되는 위치에 존재할 경우, quasi-natural gap cost를 이용하면 갭의 수를 하나 더 계산하는 착오가 발생한다. 이 논문에서는 이러한 잘못된 계산을 수정하는 방법을 제안하였다.

## 3. Dynamic programming의 결과 서열정렬을 정제(refine)하는 알고리즘

이 장에서는 MSA가 제공하는 $Opt_{quasi\text{-}natural}$을 가지는 서열정렬 중 완전히 포함되는 형태의 갭을 가질 수 있는 부분정렬만을 선택하고, natural gap cost의 갭 비용함수를 적용하여 $Opt_{natural}$을 가지는 서열 정렬을 만드는 Sub-alignment Refinement Algorithm(SRA)에 대하여 설명한다.

### 3.1. Realignment

MSA에 의해 얻을 수 있는 서열정렬은 최적의 서열정렬과 매우 유사하다. 따라서 MSA가 만들어 낸 서열정렬에서 널이 위치한 부분정렬에 대해 최적의 정렬을 만들 수 있다면 쉽게 전체 서열을 최적 혹은 최적에 가깝게 정렬할 수 있다. 이 때 정제(refine)를 시도할 부분정렬은 다음 조건 (1)과 (2)를 만족하여야 한다.

(1) 갭이 존재하는 서열의 수가 2이상이어야 한다.
(2) 가장 많은 널의 개수와 가장 적은 널의 개수의 차가 2이상이어야 한다.

조건 (1)과 (2)는 갭들이 완전히 포함되기 위한 최소한의 조건이다.

이 논문에서는 부분 서열정렬의 각 서열에 많아야 하나의 갭을 가지는 부분 서열을 만들고 비용을 계산하였다. 먼저 부분 서열의 개수를 $n$, 길이를 $l$, $i$번째 서열에 포함되는 갭의 길이를 $G_i$라고 하면, 만들어질 수 있는 가능한 부분 서열의 개수 $K$는 다음 식과 같다.

$$K = \Pi_{i=1}^{n}(l - G_i + 1)$$

새로이 만들어낸 부분정렬들에 대하여 natural gap cost를 적용하여 다시 계산한다. 이때 기존의 $Opt_{quasi\text{-}natural}$보다 작은 값을 가지는 부분정렬이 발견되면 기존의 부분정렬과 교체한다. 이때 새로운 부분정렬에 완전히 포함된 갭의 개수를 $k_1$, 하나의 갭에 대한 패널티를 $g$라고 하면 교체함에 의해 얻게되는 $Opt_{natural}$값은 다음의 보조정리 1과 같다.

**보조정리1**: $Opt_{quasi-natural} - k_1 \cdot g \leq Opt_{natural} \leq Opt_{quasi-natural}$

**증명** : 갭의 개수가 $k_1$개이므로 줄일 수 있는 비용은 최대 $k_1 * g$이 된다. 따라서 새로운 부분정렬의 값은 $Opt_{quasi-natural} - k_1 \cdot g$가 된다.□

### 3.2. Recalculation

만일 MSA에 입력으로 제공되는 정렬에 완전히 포함된 갭들이 $k_2$만큼 포함되어 있다면, 결과로 계산된 점수에는 $k_2$개에 해당되는 갭의 패널티가 더 부가되어 있는 셈이 된다. 따라서 이러한 경우 natural gap cost를 적용하기 위해 해당 갭의 개수만큼의 패널티를 감하여야 한다. 이때 새로이 얻게되는 비용은 보조정리2와 같다.

**보조정리 2**: $Opt_{natural} = Opt_{quasi-natural} - k_2 \cdot g$

**증명** : 완전히 포함되는 갭의 개수가 $k_2$개이므로 $k_2 \cdot g$만큼 더 부가된 비용을 제거하면 $Opt_{natural}$을 얻을 수 있다. □

위 보조정리 1과 2를 사용하여 다음과 같은 정리를 얻을 수 있다.

**정리** : $Opt_{quasi-natural} - k_1 g - k_2 g \leq Opt_{natural} \leq Opt_{quasi-natural}$

**증명** : 보조정리1과 보조정리2에 의해 자명하다. □

위 정리는 제안한 방법으로 줄일 수 있는 $Opt_{natural}$의 하한값을 나타낸다.

### 3.3. 알고리즘의 분석

[과정1] MSA에서 획득된 서열정렬을 잘 보존된 지역을 이용하여 몇 개의 부분정렬로 나눈다.
[과정2] 나누어진 부분에 대하여 개선될 수 있는 지역을 찾는다.
[과정3] 발견된 지역에 대해 realignment, recalculation 방법을 적용한다.
[과정4] 더 낮은 비용을 가지는 부분정렬을 얻게 되면 기존의 부분정렬과 교체한다.

그림 1 알고리즘

이 논문에서 제안한 알고리즘을 요약하면 그림 1과 같다. 이 위 알고리즘의 시간 복잡도는 [과정 3]에 의존하게 된다. 하나의 부분정렬의 비용을 계산하는데 $O(n^2)$이 필요하므로, $k$개의 서열정렬에 대해서는 $O(kn^2)$의 시간이 필요하다. 이때 공간복잡도는 서열을 저장하기 위한 공간이외에 필요한 공간이 없으므로 [과정 3]의 공간복잡도는 $O(kn)$이다.

## 4. 실험결과

이 논문의 알고리즘을 linux상에서 cc -O 명령어를 사용하여 프로그램을 컴파일하였다. 메인 메모리의 용량은 256M였다. 이 알고리즘은 많은 메모리 용량을 필요로 하지 않는다. 이 알고리즘을 테스트하기 위해 chymotrypsin, trypsin과 elastase family들로부터 단백질 서열들을 선택하여 4개의 서열들로 정렬을 만들고 개선시킬 수 있는 부분을 개선해보았다. 이때 실험에 사용된 갭 페널티는 8이었으며, 모든 서열의 비교와 관련하여 가중치 $weight(i,j)=1$로 설정하여 비용을 계산하였다.

그림 2(a)는 4개의 서열들을 MSA를 사용하여 만들어낸 정렬이며 2(b)는 본 연구에서 제안한 알고리즘을 사용하여 개선한 정렬이다.

```
CGPCEPA----SCPPLPPLGCLLGETRDAQGCCPMCARGEG
CAPCSAERMALCPPV-PASCPELTRSAGCGCCPMCALPLG
CPGCGQGVQAGCPGGCVEEEDGGSPAEGCAEAEGCLRREG
CPGCGPG----------VQEEDAGSPADGCAETGGCFRREG
                  (a)

CGPCEPAS----CPPLPPLGCLLGETRDAQGCCPMCARGEG
CAPCSAERMALCPPV-PASCPELTRSAGCGCCPMCALPLG
CPGCGQGVQAGCPGGCVEEEDGGSPAEGCAEAEGCLRREG
CPGCGPG----------VQEEDAGSPADGCAETGGCFRREG
                  (b)
```

그림 2 (a) MSA에 의해 생성된 정렬
(b) 부분정렬개선방법에 의한 정렬

MSA에 의해 제공된 정렬의 비용은 3473이었으며, 개선된 정렬은 3472로 1점이 작았다. (a)에서 없었던 완전히 포함된 갭이 (b)에 하나 존재한다.

```
CRSAYG---NELVANEEICAG    CRSAYGNE---LVANEEICAG
CKKSYP---GQITSNMFCLG     CKKSYPGQ---ITSNMFCLG
CRRRVN----------VCTL     CRRRVN----------VCTL
CLDTLHQQKQETRINIMCIG     CLDTLHQQKQETRINIMCIG
```

그림 3 (a) MSA에 의한 정렬
(b)제안한 방법에 의해 개선된 정렬

그림 3에서 MSA에서 만들어낸 정렬의 비용은 1651이었으며, 개선된 정렬은 1645였다. 이때 3(a)에서는 없었던 완전히 포함된 갭이 3(b)에는 2개가 존재한다. 이 2개의 완전히 포함된 갭이 비용을 줄이는 요인이다.

```
YYG-----YY
YYGG----YYY
YYGGGGXYYYY
```

그림 4 Recalculation

그림 4는 MSA가 만들어낸 정렬로 계산한 비용은 315였으나, 제안된 알고리즘은 동일한 정렬에 대하여 307로 더 좋은 값을 얻을 수 있었다.

개선된 프로그램의 수행시간은 갭의 개수와, 입력서열의 길이에 비례한다. 위의 예에서 걸린 시간은 모두 수초이내였다. 그림 2,3,4는 제안된 알고리즘이 효율적으로 $Opt_{natural}$을 제공함을 보여준다.

이 논문에서 제안한 알고리즘은 개선되는 부분정렬내의 서열들이 하나의 갭을 가지는 경우에만 적용할 수 있다. 만일 부분정렬내의 서열들이 하나 이상의 갭을 가질 경우, 계산량이 많아지게 되는데, 이런 경우는 매우 드물어 여기서는 고려하지 않았다. 이것의 대안으로 simulated annealing[14]과 같은 방법을 사용할 수도 있다.

## 5. 결론 및 향후과제

이 논문에서는 복수서열정렬의 최적화 기법에 대하여 논의하였다. Dynamic programming 알고리즘은 최적의 복수서열정렬을 제공하는 가장 널리 사용되는 알고리즘이나, 속성상 natural gap cost를 적용할 수 없는 것이 가장 결정적인 문제점이었다. 이러한 문제점을 해결하기 위한 다양한 방법이 제안되었지만, 어느 논문에서도 $Opt_{natural}$과 $Opt_{quasi-natural}$의 관계에 대한 이론적인 근거를 제시하지 못하였다. 이 논문에서 제안하는 개선 알고리즘을 사용하여 natural gap cost를 사용한 최적의 복수서열정렬을 얻을 수 있었으며, $Opt_{natural}$의 하한값에 대한 이론적인 근거를 제시하였다. 이 논문에서 제공한 알고리즘을 사용하면 손쉽게 MSA의 서열정렬을 개선한 최적의 서열정렬을 얻을 수 있었다. 앞으로 이 논문에서 구현한 알고리즘은 물론 복수서열정렬과 관련된 부가적인 기능을 제공하는 소프트웨어를 Visual C++로 구현하여 사용자가 보다 용이하기 사용할 수 있도록 할 계획이다.

## 참고문헌


[1] Needleman. S. B and Wunch, C. D. "A general method applicable to the search for similarities in the amino acid sequence of two proteins" *J. Molec. Biol.*, Vol. 48, pp. 443-453, 1970.

[2] Feng, D. F and Johnson, M. S. and Doolittle, R. F. "Aligning amino acid sequences: comparison of commonly used methods," *J. Molec. Evol.*, Vol.21, pp. 112-125, 1982.

[3] Ficket, J. W. "Fast optimal alignment," *Nucl. Acids Res.*, Vol.12 pp.175-180, 1984.

[4] Sankoff, D. and Kruskal, J. B. "Time Warps, String Edits and Macromolecules: The theory and practice of Sequence Comparison," *Addison-Wesley, Reading, MA*, 1983.

[5] Taylor, W. R. "Multiple sequence alignment by a pairwise algorithm," *CABIOS*, Vol.3, pp.81-87, 1987.

[6] Martinez, H. M. "A flexible multiple sequence alignment program," *Nucl. Acids. Res.*, Vol16, pp.1683-1691, 1988.

[7] Sankoff, D. "Simultaneous comparison of three or more sequences related by a tree," *Addison-Wesley, Reading, MA*, 1983.

[8] Notredame, C. and Higgins, D. "SAGA:sequence alignment by genetic algorithm," *Nucl. Acids. Res.*, Vol.24, No.8, pp.1515-1524, 1996

[9] Chan, S. C. C. and Wong, A. K. and Chiu, D. K. Y. "A survey of multiple sequence comparison methods," *Bull. Math. Bio.*, Vol.43, pp.563-598, 1992

[10] Feng, D. F. and Doolittle, R. F. "Progressive sequence alignment as a prerequisite to correct phylogenetic trees," *J. Molec. Evol.*, 25:351-360, 1987.

[11] Murata, M. and Richardson, J. S and Sussman. J. L. "Simultaneous comparison of three protein sequences," In Proc. *Natl. Acad. Sci.* USA., Vol.82, pp.3073-3077, 1985.

[12] Altschul, S. F. and Lipman, D. J. "Threes, stars, and multiple biological sequence alignment," *SIAM J. appl. Math.*, Vol.49 pp.197-209, 1989.

[13] Kim, J. and Pramanik, S. "An efficient method for multiple sequence alignment," *In Second International Conference on Intelligent Systems for Molecular Biology*, 1994.

[14] Kim, J. and Pramanik, S. and M. J. Chung. "Multiple sequence alignment using simulated annealing," *CABIOS*, Vol.10, pp.419-426, 1994.

[15] Lipman, D. J. and Altschul, S. F. and Kececioglu, J. D. "A tool for multiple sequence alignment," *Proc. Natl. Acad. Sci. USA.*, Vol.86, pp. 4412-4415, 1989.

[16]Altschul, S. F. "Gap costs for multiple sequence alignment," *J Theor. Biol.*, Vol.138 pp.297-309, 1989.

[17] Dayhoff. M. O. "A model of evolutionary change in proteins. matrices for detecting distance relationships," *In Atlas of Protein sequence an Structure*, Vol. 5 suppl.3, pp.354-352. Dayhoff. M. O.(ed) Washington. DC: National Biomedical Research Foundation, 1978.