Banded Smith-Waterman 알고리즘을 이용하여 정규화된 부분배치를 찾는 새로운 알고리즘

김상태[1] 심정섭[1] 박희진[1] 박근수[1] 박현석[2] 서정선[2]
[1]서울대학교 전기컴퓨터공학부, [2]㈜마크로젠
{stkim, jssim, hjpark, kpark}@theory.snu.ac.kr, {hspark, jeongsun}@macrogen.co.kr

A new algorithm for finding normalized local alignment using banded Smith-Waterman algorithm

Sangtae Kim[1] Jeong Seop Sim[1] Heejin Park[1] Kunsoo Park[1]
Hyunseok Park[2] Jeong-Sun Seo[2]
[1]School of Electrical Engineering and Computer Science, Seoul National University
[2]Macrogen

요 약

두 문자열의 부분배치(local alignment)를 찾는 대표적인 알고리즘인 Smith-Waterman 알고리즘(SW 알고리즘)은 정규화된 최적부분배치를 찾지 못하는 단점이 있다. 최근에 fractional programming 기법을 이용하여 여러 번의 SW 알고리즘을 수행함으로써 정규화된 최적부분배치를 찾는 알고리즘이 제시되었지만 이는 매우 많은 시간이 걸린다. 본 논문에서는 fractional programming 기법을 이용하여 정규화된 최적부분배치를 찾는 알고리즘에, 완전매치(Exact Match)를 이용한 휴리스틱 기법인 Banded SW 알고리즘을 적용하여, 낮은 오차를 가지면서 실용적으로는 매우 빠른 정규화된 최적부분배치를 찾는 알고리즘을 제시하고 이 알고리즘과 기존의 알고리즘을 직접 구현하여 실험한 결과를 비교 분석한다.

1. 서 론

두 서열(sequence) a, b가 주어졌을 때, a, b의 최적부분배치를 찾는 문제는 각 문자열에서 유사도가 가장 높은 부분을 찾아내는 문제이다. 이 문제는 생물정보학에서 매우 중요한 문제로, 유전자나 단백질 서열 간의 유사성 비교, 서열에 기반한 단백질의 구조 예측, fragment assembly 등의 분야에 광범위하게 응용되고 있다. 특히 최근에는 여러 생물의 전체 유전체(genome) 서열이 밝혀짐에 따라, 최적부분배치(optimal local alignment)를 찾는 문제를 이용하여 유전자 예측 (gene prediction) 문제를 해결하는 새로운 방법이 제시되었다[2].

최적부분배치를 찾는 대표적인 알고리즘은 Smith-Waterman 알고리즘[3](SW 알고리즘)이다. SW 알고리즘은 동적프로그래밍(dynamic programming) 기법을 이용하여 $O(n^2)$ 시간에 최적부분배치를 찾는다. 그러나, SW 알고리즘은 부분배치를 구성하는 문자열의 길이를 고려하지 않음으로 인해, 찾아낸 부분배치의 내부에 유사도가 매우 낮은 부분을 포함할 수 있다. 또한, 생물학적으로는 길이에 비해 유사도가 높은 부분배치가 중요하게 고려됨에도 불구하고 이를 찾지 못할 수 있다. 이러한 문제들 때문에 유전자 예측 분야에 SW 알고리즘을 적용했을 때, 원하는 결과를 찾을 수 없는 경우가 발생할 수 있다[4].

Arslan 등이 이러한 단점을 해결하는 방법을 제시하였다. Arslan 등은 fractional programming 기법을 적용하여 SW 알고리즘을 여러 번 수행하여 정규화된, 즉 부분배치를 구성하는 문자열의 길이를 고려한 최적부분배치를 찾는 알고리즘을 제시하였다[1]. 이 알고리즘은 이론적으로 $O(n^2 \log n)$ 시간이 걸리며 SW 알고리즘보다 3-5배 느리게 된다.

본 논문에서는 Arslan 등이 제시한 정규화된 최적부분배치를 찾는 알고리즘에, 완전매치(Exact Match)를 이용하여 SW 알고리즘을 빠르게 수행하는 Banded SW 알고리즘을 적절히 적용하여 실용적으로 훨씬 빠른 정규화된 최적부분배치를 찾는 알고리즘을 제시한다. 또한, Arslan 등이 제시한 알고리즘과 본 논문에서 제시하는 알고리즘을 실제로 구현하고 여러 데이터를 이용하여 실험해본 결과를 비교 분석한다. 본 논문에서 제시하는 알고리즘은 최적인 부분배치를 찾지 못할 수도 있지만 실험 결과에 의하면 오차율은 매우 작다.

2장에서는 관련된 이전 연구에 대해 살펴보고, 3장에서는 Banded SW 알고리즘을 이용하여 정규화된 최적부분배치를 찾는 새로운 알고리즘을 제시한다. 4장에서는 우리가 제시한 알고리즘과 Arslan 등이 제시한 알고리즘을 구현하여 실험한 결과를 밝히고 성능을 비교하며 5장에서 결론을 맺는다.

2. 이전 연구

2.1 배치와 유사도

알파벳 집합 Σ상의 문자들로 이루어진 두 서열(Sequence) $a = a_1 a_2 \dots a_n$, $b = b_1 b_2 \dots b_m$에 공백(Δ)을 삽입하여 한 문자열의 임의의 문자(또는 공백)가 다른 문자열의 같은 위치의 문자(또는 공백)에 대응되도록, $a^* = a_1^* a_2^* \dots a_L^*$, $b^* = b_1^* b_2^* \dots b_L^*$ ($a_i, b_i \in \Sigma \cup \{\Delta\}$, $1 \le i \le L$)를 만들었을 때, a^*, b^*를 a, b의 배치(alignment)라고 한다.

예제 1. a = AAACTG 이고 b = CAAGCTA 일 때,

a^* = A A A Δ C T G

b^* = C A A G C T A 는 a, b의 배치이다.

a, b의 배치 a^*, b^*에서 $a_i^* = b_i^* \neq \Delta$일 때 이를 일치(match), $(a_i^* \neq b_i^*) \wedge (a_i^*, b_i^* \neq \Delta)$일 때 이를 불일치(mismatch), a_i^* 또는 b_i^* 중 하나만이 Δ일 때 이를 갭(gap)이라고 한다. 예제 1에서, $a_2^* = b_2^* =$ A 이므로 이는 일치이고, $a_1^* =$ A, $b_1^* =$ C이므로 이는 불일치이며, $a_4^* = \Delta$, $b_4^* =$ G이므로 이는 갭이다.

두 배치 a^*, b^*의 배치유사도란, 일치의 개수, 불일치의 개수, 갭의 개수를 각각 x, y, z라 하고, 일치 점수를 1, 불일치 벌점(penalty)을 δ, 갭 벌점을 μ라고 할 때, $x - \delta y - \mu z$의

값으로 정의된다. 두 서열 a, b의 유사도 $S(a,b)$는 a, b의 가능한 모든 배치 중 유사도가 가장 높은 배치의 배치유사도로 정의된다. 불일치 벌점이 0.7, 갭 벌점이 1일 때, 예제 1의 유사도는 4-0.2×2-1×1 = 1.6이 된다.

2.2 최적부분배치

서열 α가 서열 β의 부분문자열일 때 $\alpha < \beta$로 나타내기로 하고 문자열 a의 길이를 $|a|$로 나타내기로 하자. 두 서열 $a = a_1a_2...a_m$, $b = b_1b_2...b_n$가 주어졌을 때, a, b의 모든 부분문자열 $\alpha(<a)$, $\beta(<b)$중 $S(\alpha,\beta)$의 값이 가장 큰 α, β의 배치를 a, b의 **최적부분배치**라고 하고 a, b의 최적부분배치의 유사도를 LA(a, b)로 표시한다.

최적부분배치를 찾는 알고리즘으로 가장 널리 알려진 것이 Smith-Waterman 알고리즘[3](SW 알고리즘)이다. SW 알고리즘은 두 서열 a, b에 대해, $a = a_1a_2...a_i$, $b = b_1b_2...b_j$의 부분배치의 유사도의 최대값을 $H_{i,j}$라고 할 때, $H_{i,j}$는 동적프로그래밍 기법을 이용하여 $(m+1)(n+1)$ 크기의 테이블(SW 테이블)을 만들어 구할 수 있다. 즉, $H_{i,j}$ $(0 \le i \le n,\ 0 \le j \le m)$는 아래의 식 (1)을 이용하여 구할 수 있다.

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

$$H_{i,j} = \max\{0,\ H_{i-1,j-1} + s(a_i, b_i),\ H_{i,j-1} - \mu\}$$

$$s(a_i, b_i) = \begin{cases} 1 & \text{if } a_i = b_i \\ -\delta & \text{if } a_i \neq b_i \end{cases} \qquad (1)$$

이 때, 모든 $H_{i,j}$중 최대값이 $LA(a,b)$가 되며 $O(mn)$ 개의 $H_{i,j}$값을 계산해야 하므로 수행시간은 $O(mn)$이 된다. 그림 1은 예제 1에 대한 SW 테이블을 나타낸다.

	Δ	C	A	A	G	C	T	A
Δ	0	0	0	0	0	0	0	0
A	0	0	1	1	0	0	0	1
A	0	0	1	2	1	0	0	1
A	0	0	1	2	1.3	0.3	0	1
C	0	1	0	1	1.3	2.3	1.3	0.3
T	0	0	0.3	0	0.3	1.3	3.3	2.3
G	0	0	0	0	1	0.3	0.6	2.6

그림 1. SW 테이블

2.3 정규화된 최적부분배치

두 서열 a, β에 대해 $S(a,\beta)/(|a|+|\beta|+L)$의 값($L$은 상수)을 a, β의 정규화된 유사도라고 한다. 주어진 두 서열 a, b에 대해, a, b의 부분문자열들 중에서 정규화된 유사도가 가장 큰 부분문자열 $\alpha(<a)$, $\beta(<b)$를 a, b의 최적부분(optimal segment)이라고 하고, α, β의 배치를 a, b의 **정규화된 최적부분배치**라고 한다. 두 서열 a, b에 대해, 일치, 불일치, 갭의 개수가 각각 x, y, z인 부분배치가 존재한다면, (x,y,z)를 a, b의 배치벡터라고 한다. 두 서열 a, b의 모든 가능한 배치벡터의 집합을 AV(a, b)라고 한다.

$(x,y,z) \in$ AV(a, b)일 때, 서열 a, b의 최적부분배치의 유사도를 찾는 문제(LA)와 정규화된 최적부분배치의 유사도를 찾는 문제(NLA)를 아래와 같이 정의한다.

LA : $(x,y,z) \in$ AV(a, b)인 모든 (x,y,z)에 대해, $x - \delta y - \mu z$의 최대값을 찾는 문제

NLA : $(x,y,z) \in$ AV(a, b)인 모든 (x,y,z)에 대해, $\dfrac{x - \delta y - \mu z}{2x+2y+z+L}$ 의 최대값을 찾는 문제

한편, Fractional Programming 기법을 이용하기 위해 매개변수 최적부분배치의 유사도를 찾는 문제를 아래와 같이 정의한다.

LA(λ) : $(x,y,z) \in$ AV(a, b)인 모든 (x,y,z)에 대해, $x - \delta y - \mu z - \lambda(2x+2y+z+L)$의 최대값을 찾는 문제

이 때, $\lambda < 1/2$이면 LA(λ) 문제와 LA문제는 δ, μ를 변화시킴으로써 상수 시간에 서로 변환 가능하므로, LA문제를 해결하는 SW 알고리즘을 이용하여 LA(λ) 문제의 최적해를 구할 수 있다. 또한 LA(λ) 문제의 최적해가 0일 때 λ는 NLA의 최적해이므로, SW 알고리즘을 재귀적으로 호출하여 LA(λ) 문제의 최적해가 0이 되도록 λ값을 변화시키면 NLA문제의 최적해를 구할 수 있다[1].

다음은 위의 사실을 이용하여 최적부분배치의 유사도를 찾는 알고리즘이다.

알고리즘 Dinkelbach

1. $(x,y,z) \in$ AV(a, b)인 임의의 배치벡터 (x,y,z)를 선택
2. $\lambda^* = \dfrac{x - \delta y - \mu z}{2x+2y+z+L}$
3. **repeat**
4. $\quad \lambda = \lambda^*$
5. $\quad$ LA(λ) 문제를 LA문제로 변환 후 SW 알고리즘을 적용하여 배치벡터 (x,y,z)를 얻어냄
6. $\quad \lambda^* = \dfrac{x - \delta y - \mu z}{2x+2y+z+L}$
7. **until** $\lambda = \lambda^*$
8. λ^*를 출력

알고리즘 Dinkelbach는 평균적으로 3-5번의 SW 알고리즘을 수행한다. 한편, [1]에서는 δ, μ가 유리수일 때, $O(n^2 \log n)$시간에 NLA문제를 해결하는 알고리즘도 제시하고 있으나 실용적으로는 알고리즘 Dinkelbach가 더 빠르다.

3. Banded Smith-Waterman 알고리즘을 이용한 새로운 알고리즘

3.1 Banded SW 알고리즘

두 서열의 최적부분배치는 내부에 일정 길이 이상의 연속된 일치를 포함하고 있을 확률이 높다. Banded SW은 이 사실을 이용하여 두 서열에서 완전일치를 찾은 후 SW 테이블의 전체를 계산하지 않고, 완전일치가 일어난 주변의 일부 값들만을 계산한다. Phrap[5], STROLL[6], FASTA[7] 등의 프로그램에서 응용에 맞게 이를 적절히 적용하고 있다. Banded SW 알고리즘은 다음과 같이 두 단계로 구성된다.

1) 밴드(band)의 정의

먼저, 두 서열 a, b사이의 주어진 길이 이상의 완전일치를 모두 찾는다. a, b에서 찾은 완전일치가 각각 $a_i...a_j$, $b_i...b_j$ 일 때, $i-j$를 완전일치의 대각선이라고 정의한다. 다음으로 완전일치를 대각선의 순서대로 정렬한다. 완전일치가 발생한 대각선 d에 대해 구간 $[d-k,\ d+k]$를 밴드로 정의한다. (k는 입력으로 주어진 양의 정수) 만약 밴드가 겹치면, 겹치는 밴드들을 합쳐서 하나의 큰 밴드로 만든다. 구간 $[p,\ q]$인 밴드에 대해 밴드폭(*bandwidth*)을 $q-p+1$로 정의한다.

2) 정해진 밴드에 대해 SW 알고리즘 수행

이 과정은 SW 알고리즘과 같은 방식으로 진행한다. 단 $H_{i,j}$를 구할 때, $i-j$가 위에서 구한 밴드에 속하는 $H_{i,j}$들의 값만 계산하며, $i-j$가 위에서 구한 밴드에 속하지 않는다면 0으로 가정한다.

3.2 Banded SW 알고리즘을 적용하여 정규화된 부분배치를 찾는 새로운 알고리즘

알고리즘 Dinkelbach는 두 서열 a, b에 대해 정규화된 최적부분배치의 유사도를 찾는다. 그러나, 두 서열 a, b의 부분 문자열 중에서 정규화된 유사도가 제한값 T이상인 부분배치를 모두 찾는 문제(ANLA)가 생물학적 응용에 더 적합하다. [1]에서는 ANLA 문제를 해결하기 위하여 a, b의 정규화된 최적부분배치를 찾은 후 a, b의 최적부분 $\alpha(<a)$, $\beta(<b)$를 가리고(masking), 다시 최적부분배치를 찾는 것을 반복하는 방법을 사용하였다.

우리는 Banded SW 알고리즘을 효율적으로 적용하여 이 문제를 해결하는 새로운 알고리즘을 제시한다.

알고리즘 FastANLA

1. 서열 a, b에서 길이가 M이상인 완전일치를 찾아 밴드를 결정
2. 모든 밴드에 대해,
3. **repeat**
4. 현재의 밴드와 a, b에 대해 SW 알고리즘을 수행하여 배치벡터 (x, y, z)를 얻음
5. $\lambda^* = \frac{x-\delta y-\mu z}{2x+2y+z+L}$
6. **repeat**
7. $\lambda = \lambda^*$
8. LA(λ) 문제를 LA 문제로 변환한 후 현재의 밴드에 대해 SW 알고리즘을 적용하여 배치벡터 (x, y, z)를 얻어내고 최적부분배치와 최적부분을 구함
9. $\lambda^* = \frac{x-\delta y-\mu z}{2x+2y+z+L}$
10. **until** $\lambda = \lambda^*$
11. **if**($\lambda \geq T$)
12. 마지막으로 구한 최적부분배치를 출력하고 a, b에서 마지막으로 구한 최적 부분을 가림
13. **until** $\lambda < T$

여러 번의 SW 알고리즘을 수행할 때, δ, μ의 값은 바뀌지만 서열 a, b는 변하지 않으므로 완전매치를 찾아 밴드를 구하는 전처리는 한 번만 수행한다. 또한 하나의 최적부분을 찾을 때, 모든 밴드에 대해 SW 알고리즘을 수행하지 않고 하나의 밴드에 대해 SW 알고리즘을 수행한다.

알고리즘 FastANLA는 [1]에서 제시한 ANLA 문제를 해결하는 알고리즘(알고리즘 ADinkelbach)보다 훨씬 빠르게 동작한다. 주어진 서열 $a(|a|=m)$, $b(|b|=n)$에 대해, 1행의 완전일치는 접미사 배열[8]을 사용하면 $O((m+n)\log(m+n))$ 시간 내에 구할 수 있다. 또한 하나의 최적부분을 찾기 위해 수행하는 SW 알고리즘의 평균 횟수를 ρ라고 할 때, k개의 최적부분을 찾기 위해 알고리즘 ADinkelbach는 $O(\rho \cdot k \cdot m \cdot n)$ 만큼의 계산을 수행하나, 알고리즘 FastANLA는 $O(\rho \cdot k \cdot bandwith \cdot n)$ 만큼의 계산을 수행하므로 알고리즘 ADinkelbach보다 $m/bandwith$배만큼 빠르다.

그러나, 최적부분 사이의 완전일치가 정해진 값 이하이거나 최적부분배치 내부에 갭의 개수가 많을 경우, 유사도가 제한값 T 이상인 최적부분 중 찾지 못하는 것이 있을 수도 있다.

4. 실험 결과 및 성능 비교

알고리즘 ADinkelbach와 FastANLA를 구현하여, 수행 시간과 정확도를 비교하였다. C++로 구현하였으며, Pentium III 866Mhz ×2 워크스테이션을 이용하여 실행하였다.

표 1은 $L = 2000$, $\delta = 1$, $\mu = 0.2$인 조건에서, human (GenBank Acc. No. AF030876, Z47046, Z47066, Z68193)과 mouse(GenBank Acc. No. AF121351)의 유전체 서열들에 대해 두 알고리즘을 적용하여 정규화된 유사도가 0.1 이상인 모든 최적부분을 찾는 실험의 결과이다. 알고리즘 FastANLA에서 M 값은 20을 사용하였으며 k 값은 50, 75, 100을 사용하였다. 입력 크기는 비교할 두 서열의 길이의 기하평균이다.

입력 크기	수행시간(초)			
	ADinkel -Bach	FastANLA $k=50$	FastANLA $k=75$	FastANLA $k=100$
10K	382	3.4	4.2	5.3
20K	1631	14	19	24
50K	43549	99	149	199
100K	383276	453	701	954

표 1. ADinkelbach와 FastNLA의 수행시간

알고리즘 FastANLA는 알고리즘 ADinkelBach보다 대략, 입력크기/($bandwidth \fallingdotseq k \times 2$)배정도 빠르다는 사실을 확인할 수 있다. 또한, k가 100, 75일 때는 알고리즘 ADinkelBach에서 찾은 모든 부분배치를 알고리즘 FastANLA에서 찾을 수 있었으며, k가 50일 때는 9%를 찾지 못하였다.

5. 결론

본 논문에서는 정규화된 유사도가 제한값 T 이상인 최적부분을 모두 찾는 문제에 Banded SW 알고리즘의 기법을 적용하여 실용적으로 매우 빠른 알고리즘을 제시하였다. 이 알고리즘의 특징은 밴드폭(k)을 조절하여 매우 정확한 결과를 얻을 수 있다는 것이다. 따라서 본 알고리즘은 유전자 예측 등의 생물학적인 문제에 효과적으로 이용될 수 있을 것이다.

6. 참고문헌

[1] A. N. Arslan, Ö. Eğecioğlu, P. Pevzner, A new approach to sequence comparison: normalized sequence alignment, Bioinformatics, 17 327-337 (2001).

[2] P. S. Novichkov, M. S. Gelfand, A. A. Mironov, Prediction of the exon-intron structure by comparison sequences, Molecular Biology, 34 200-206 (2000).

[3] M.S. Waterman, Introduction to Computational Biology, Chapman & Hall, London, (1995).

[4] N. N. Alexandrov, V. V. Solovyev, Statiscal significance of ungapped alignments, Pacific Symposium on Biocomputing(PSB-98), 463-472 (1998).

[5] P. Green, Documentation for phrap. Genome Center, University of Washington, http://www.phrap.org/phrap.docs/phrap.html.

[6] T. Chen, S. S. Skiena, Trie-Based Data Structures for Sequence Assembly, The Eighth Symbosium on Combinatorial Pattern Matching, 206-223 (1997).

[7] D. Lipman , W. Pearson, Improved tools for biological sequence comparison, Proc. of Natl. Acad. Sci., 85 2444-2448 (1988).

[8] U. Manber, G. Myers, Suffix arrays : a new method for on-line string searches, SIAM J. Comput., 935-948, (1993).