Proceedings of KIIS Spring Conference 2008 Vol. 18, No. 1.

서열의 길이에 무관한 유사도 측정 알고리즘

A Sequence Similarity Algorithm Irrelevant to Sequence Length

김재광 · 이지형
Jaekwang Kim and Jee-Hyong Lee

성균관대학교 전자전기컴퓨터공학과
E-mail: {linux, jhlee}@ece.skku.ac.kr

요 약

Dynamic Programming (DP)을 이용한 서열 비교 알고리즘은 DNA, RNA, 단백질 서열의 비교와 프로그래밍 소스 코드 유사도를 측정하는 곳 등에 널리 사용되어 왔다. 이 알고리즘은 DP를 이용하여 행렬을 구성한 후, 행렬의 가장 마지막 생성 값을 이용해 두 서열의 유사도를 측정하는 방법이다. 그러나 이 알고리즘에서 사용하는 마지막 생성 값은 비교 서열이 길이에 따라 크게 좌우되기 때문에 다양한 서열들의 유사도를 알아내기에는 부적합하다. 본 논문에서는 서열의 길이에 무관한 유사도 측정 (S2) 알고리즘을 제안한다. 제안된 알고리즘을 이용하면 비교 서열의 길이에 영향을 받지 않고 정당한 서열 비교를 할 수 있다. 제안된 알고리즘의 검증을 위해 본 논문에서는 프로그램 소스 코드의 유사도 측정을 수행한다.

키워드 : 서열의 길이에 무관, 서열 비교 알고리즘, 유사도 측정, Dynamic Programming

1. 서 론

Dynamic Programming (DP) 알고리즘은 1957년 Richard Bellman에 의해 제안된 방법으로서 최적성의 원리에 부합하는 문제 해결 방법이다[1]. 1970년 S. Needleman과 C. Wunsch는 DP를 이용하여 DNA, RNA, 단백질 서열의 비교를 위한 일반적인 알고리즘을 개발하였고, 이 알고리즘은 생물학 정보 외에도 프로그래밍 소스 코드 유사도 측정과 같은 서열 비교를 위한 유용한 알고리즘으로 널리 사용되어 왔다.

이러한 알고리즘은 생물학 정보 서열의 비교나 프로그래밍 소스 코드 유사도 등을 구할 때에는 사용되지만 비교 서열의 길이에 영향을 받는다는 문제가 있다. 구체적으로, DP를 이용하여 서열의 유사도를 구하는 과정에서 서열의 길이가 길어질수록 행렬의 크기가 커지고 행렬의 마지막 생성 값도 커지므로 정당한 서열 비교 알고리즘으로서는 치명적인 문제가 있다.

본 논문에서는 행렬의 크기에 무관한 Sequence Similarity (S^2) 알고리즘을 제안한다. 제안된 알고리즘을 이용하면 비교 서열의 길이에 영향을 받지 않고 정당한 서열 비교를 할 수 있다. 제안된 알고리즘의 검증을 위해 본 논문에서는 프로그램 소스 코드의 유사도 측정을 수행하였고, 그 결과 기존 알고리즘에 비교하여 S^2 알고리즘을 사용하였을 때, 행렬의 크기에 무관한 정당한 평가가 이뤄짐을 확인하였다.

감사의 글 : 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 지식경제부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 08B3-B1-10M 과제로 지원된 것임

본 논문은 2장에서 DP를 이용한 기존의 알고리즘을 말하고, 3장에서는 S^2 알고리즘을 제안한다. 4장에서는 알고리즘의 검증을 위한 실험과 그 결과를 보이며, 마지막 5장에서는 결론을 나타낸다.

2. 기존 알고리즘

기존 알고리즘은 DP를 이용한다. DP는 N2의 수행 시간 동안 동작하는 $O(N^2)$ 알고리즘으로서 최적화 문제의 해결을 위해 개발되었다. DP 알고리즘의 기본 개념은 다음과 같은 세 단계로 표현된다.

1. 가장 기본적인 문제의 답부터 계산
2. 순차적으로 더 큰 문제의 답을 계산
3. 전체 문제의 답을 계산

1970년에 S. Needleman 과 C. Wunsch은 DP를 이용하여 생물 정보 서열 정렬에 이용하는 알고리즘을 제안하였다[2]. 이들이 제안한 두 서열의 최적 정렬 알고리즘은 DNA, RNA 및 단백질의 서열 정렬을 위한 유용한 방법으로 널리 사용되어 왔을 뿐 아니라 최근에는 프로그램 소스 코드의 유사도를 측정하기 위한 방법으로도 사용되고 있다[3][4].

S. Needleman과 C. Wunsch이 제안한 서열 정렬을 위한 알고리즘을 이용하면 다음과 같은 방법으로 두 서열을 비교하고 최적의 정렬을 찾을 수 있다.

```
G C T G G A A G G C A T
  G C A G A G C A C T
```

그림 1. 두 DNA 서열.
Fig. 1. Two DNA sequences.

그림 1이 보이는 바와 같은 두 DNA 서열을 서로 비교하여 가장 최적의 정렬을 찾고자 할 때에는 먼저 두 개의 서열을 이용하여 그림2가 보이는 바와 같이 2차원 행렬을 구성한다[5]. 행렬을 구성할 때, 각 행렬의 값을 계산하기 위한 과정은 먼저 각 행렬의 기저 값으로부터 가장 가까이에 있는 다음 행이나 열, 혹은 대각선 위치의 값을 DP를 이용하여 구한 다음, 최종적으로 마지막 행렬의 값까지 구하는 것이다.

그림 2가 보이는 바와 같이 행렬을 구성할 때에는 가장 오른쪽 아래 값이 최종적으로 계산된다. 이 값은 행렬의 모든 구성 값 중에서 가장 큰 값이며, 다른 조건이 같을 때, 이 값이 클수록 두 서열의 유사도가 높음을 나타낸다.

		G	C	T	G	G	A	A	G	G	C	A	T
	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
G	-1	2	1	0	-1	-2	-3	-4	-5	-6	-7	-8	-9
C	-2	1	4	3	2	1	0	-1	-2	-3	-4	-5	-6
A	-3	0	3	3	2	1	3	2	1	0	-1	-2	-3
G	-4	-1	2	2	5	4	3	2	4	3	2	1	0
A	-5	-2	1	1	4	4	6	5	4	3	2	4	3
G	-6	-3	0	0	3	6	5	5	7	6	5	4	3
C	-7	-4	-1	-1	2	5	5	4	6	6	8	7	6
A	-8	-5	-2	-2	1	4	7	7	6	5	7	10	9
C	-9	-6	-3	-3	0	3	6	6	6	5	7	9	9
T	-10	-7	-4	-1	-1	2	5	5	5	5	6	8	11

그림 2. S. Needleman과 C. Wunsch이 제안한 알고리즘을 이용하여 DNA 서열 정렬을 하기 위한 행렬 생성 결과.
Fig. 2. Matrix generation result for DNA sequence alignment using the algorithm suggested by S. Needleman and C. Wunsch.

행렬을 구한 후에는 그림 3이 보이는 바와 같이 역추적 과정을 통해 최적 경로를 선택하는 과정이 수행된다. 이를 통해 두 DNA 서열은 그림 4가 보이는 예와 같이 몇 가지 최적 정렬을 수행한다.

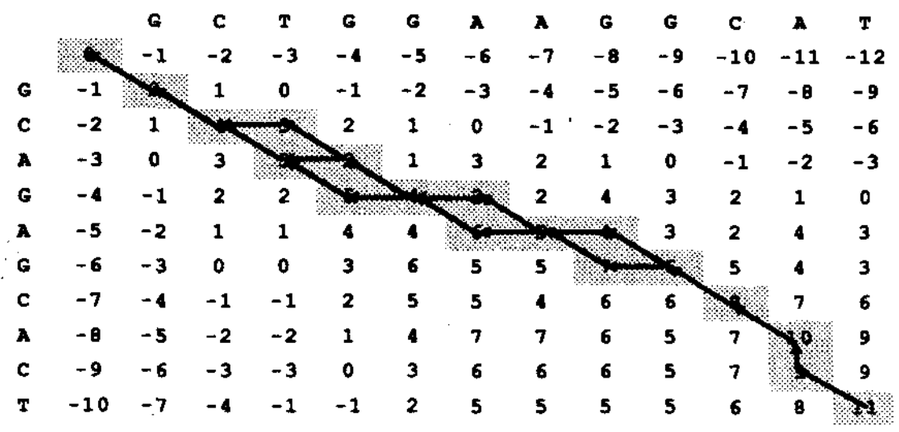

그림 3. S. Needleman과 C. Wunsch이 제안한 알고리즘을 이용하여 생성된 행렬에서 최적 경로를 선택하는 역추적 과정.
Fig. 3. Backtracing for finding optimal path in the generated matrix using the algorithm suggested by S. Needleman and C. Wunsch.

```
G C T G G A A G G C A - T
G C A G - A G - C - A C T

G C T G G A A G G C A - T
G C - A G - A - G C A C T

G C T G G A A G G C A - T
G C A - G A - G - C A C T
        .
        .
        .
```

그림 4. S. Needleman과 C. Wunsch이 제안한 DP를 이용한 두 DNA 서열의 정렬의 예.
Fig. 4. The example of the sequence alignment of two DNA sequences suggested by S. Needleman and C. Wunsch.

3. S^2 알고리즘

S. Needleman과 C. Wunsch이 제안한 DP를 이용한 알고리즘을 통해 두 서열의 최적 정렬을 구하기 위해서는 행렬을 구성한 후, 역추적 과정을 통해 정렬을 하는 것이 필요하다. 반면, 두 서열의 유사도만을 비교할 때에는 행렬을 구성하여 행렬의 가장 마지막 생성 값만을 이용할 수 있다. 즉 조건이 같을 경우, DP를 사용하여 구성한 행렬의 최종 생성 값이 클수록 두 서열의 유사도가 높다. 그런데 행렬의 최종 생성 값은 서열의 길이에 따라 민감하게 달라지기 때문에, 다양한 길이의 서열 간 유사도를 비교하는 데에는 문제가 있다.

본 논문에서는 이 문제를 해결하고자 Sequence Similarity (S^2) 알고리즘을 제안한다.

3.1 S^2 알고리즘

S^2 알고리즘은 두 서열의 유사도를 비교할 때, 행렬의 크기에 무관한 결과를 산출하도록 정규화(Normalized) 과정을 수행한다. 이때 정규화의 과정은 다음과 같은 식 1에 기준한다.

$$V_{nor} = \frac{V_{\max}}{2}\left(\frac{SL_A + SL_B}{SL_A \times SL_B}\right) \quad (1)$$

식 1이 보이는 바와 같이 정규화 과정은 두 서열의 길이에 대하여 이뤄진다. 식 1에서 V_{nor} 값은 정규화된 값이고, $V_{\max}$ 값은 DP를 이용하여 구성된 행렬에서 가장 마지막에 생성되는 값이며, SL_A는 서열 A의 길이로서 행렬의 한쪽 행을 이루는 수이고, SL_B는 서열 B의 길이로서 행렬의 한쪽 열을 이루는 수이다.

식 1에 의하여 $V_{\max}$ 값은 V_{nor} 값으로 정규화 된다. 정규화된 값인 V_{nor} 값은 소스 코드 알고리즘을 수행한 결과로서 두 서열의 유사도를 비교할 때, 행렬 길이와

무관한 정당한 비교를 가능하게 한다.

3.2 S^2 알고리즘의 성능

S^2 알고리즘은 기본적으로 DP를 이용하기 때문에 N2의 수행 시간 동안 동작하는 $O(N^2)$ 알고리즘이다. S^2 알고리즘은 수식 상으로 DP와 동일한 복잡도를 가지고 있다.

4. 실험 및 결과

본 절에서는 Control Flow에 기반한 프로그램 소스 코드의 유사도를 비교하는 실험에 기존 알고리즘과 S^2 알고리즘을 이용하여 클러스터링 하였을 때, 제안된 S^2 알고리즘의 유용성을 보이는 실험과 결과를 기술한다.

4.1 실험 환경

실험에 사용한 데이터는 파이썬 프로그램 소스 코드 13개이다. 실험 데이터에 관한 설명은 표 1이 보이는 바와 같다.

표 1이 보이는 바와 같이 (A) 그룹은 p1.py ~ p5.py, 그리고 (B) 그룹은 p5_nor.py. (C) 그룹은 dp1.py ~ dp4.py, (D) 그룹은 test1.py ~ test3.py으로 이루어져 있다. 각 그룹은 비슷한 프로그램 소스 파일이다. 각 파일의 이름이 같고 뒤에 있는 숫자가 높을수록 새롭게 기능이 추가되거나, 수정이 되었다는 것을 의미한다.

(B) 그룹에 속한 p5_nor.py 파일은 p5.py 파일을 수정하여 생성하였다.

한편, (C) 그룹의 내용은 (A)와 (B) 그룹의 프로그램 소스 코드에 포함되어 있으므로 상당한 유사도를 가질 것으로 예상된다. 반면 (D) 그룹의 프로그램 소스 코드들은 단백질 구조 분석을 위한 프로그램으로서 다른 그룹의 데이터들과 다른 형태를 가질 것으로 예상된다.

표 1. 실험 데이터.
Table. 1. Test data.

파일 이름 (그룹)	파일 내용	크기 (byte)
p1.py ~ p5.py (A)	DP기반의 기존 알고리즘을 이용한 소스 코드 비교 프로그램	5.4K~ 9.0K
p5_nor.py (B)	소스 코드 알고리즘을 이용한 소스 코드 비교 프로그램	9.1K
dp1.py ~ dp4.py (C)	DP를 구현한 프로그램	2.8K~ 3.1K
test1.py ~ test3.py (D)	단백질 구조를 분석하기 위한 프로그램	13K

각 실험 데이터에 대해 기존의 알고리즘과 제안된 알고리즘을 이용하여 유사도 값을 구하였으며, 그 유사도 값을 입력으로 트리 형태의 클러스터링을 수행하였다. 클러스터링 도구로는 oc 라는 오픈 소스 프로그램을 사용하였다[6].

4.2 실험 결과

실험 결과 그림 5와 그림 6이 보이는 바와 같은 트리 형태의 클러스터링이 수행되었다.

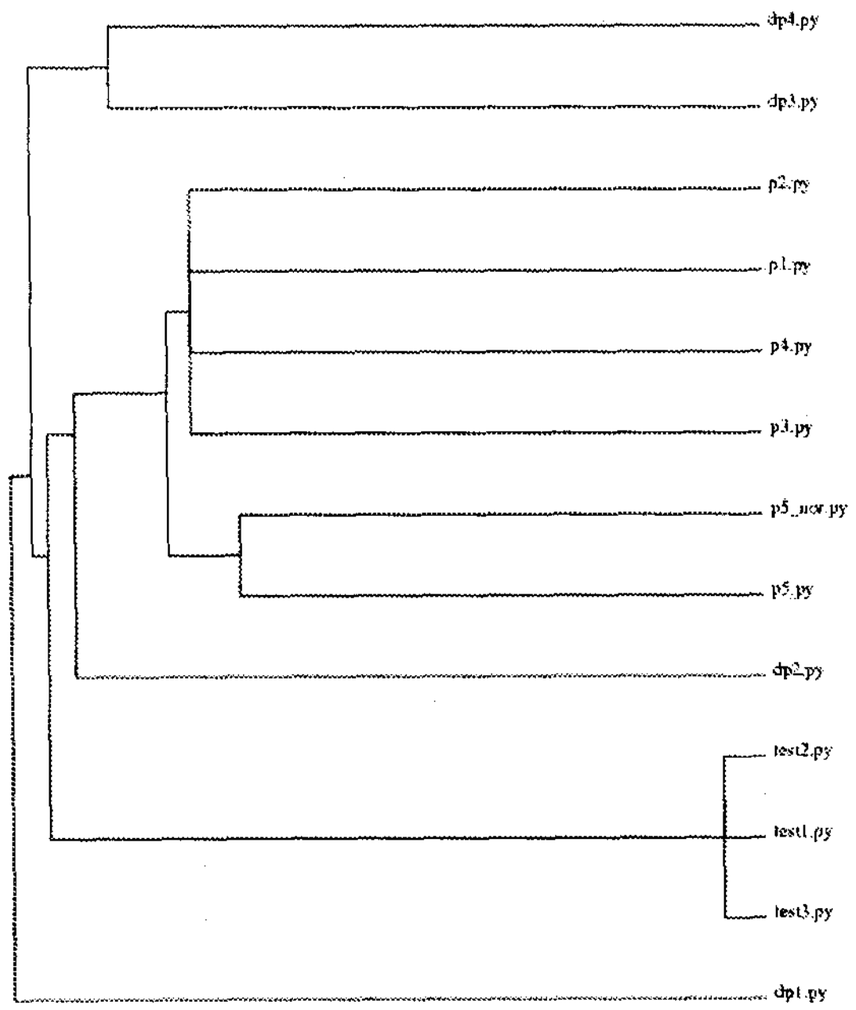

그림 5. DP기반의 기존 알고리즘을 이용한 Control Flow 기반 프로그램 소스 코드 비교.
Fig. 5. The comparison of the program source code based on control flow using DP based algorithm.

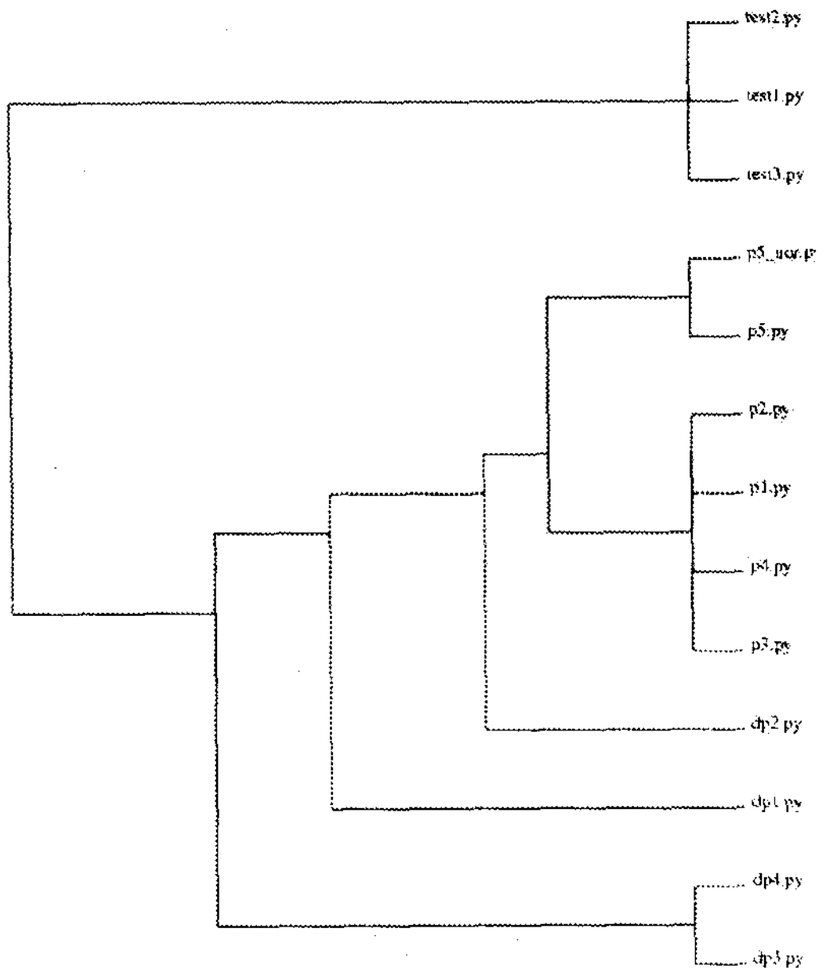

그림 6. S^2 알고리즘을 이용한 Control Flow 기반 프로그램 소스 코드 비교
Fig. 6. The comparison of the program source code based on control flow using S^2 algorithm.

그림 5는 DP기반의 기존 알고리즘을 이용하여 13개 데이터 파일에 대하여 Control Flow에 기반한 유사도를 측정하고, 클러스터링을 한 결과를 보인다. 그림 5에서 보이는 바와 같이 기존 알고리즘을 사용하였을 경우에는 (C) 그룹의 구성 파일들이 모두 제각각 흩어지고 클러스터링 되지 않은 것을 알 수 있다.

반면 S^2 알고리즘을 이용하여 각 파일의 Control Flow에 기반한 유사도를 측정 후, 클러스터링 한 결과를 살펴보면, 그림 6이 보이는 바와 같이 가장 유사도가 떨어지는 (D) 그룹이 따로 분류된 것을 확인할 수 있으며, (A), (B), (C) 그룹을 모두 적절히 분류한 것이 확인된다.

결과를 통해 알 수 있듯이 S^2 알고리즘을 이용하면 행렬의 생성 크기에 큰 영향을 받지 않고, 정당한 서열 비교를 할 수 있다.

5. 결 론

S^2 알고리즘은 두 서열의 유사도를 비교할 때, 행렬의 크기에 무관한 결과를 산출하도록 정규화 (Normalized) 과정을 수행하는 알고리즘이다. 이를 통해 길이가 다른 여러 서열 간에 유사도를 비교할 때, 행렬의 마지막 생성 값만을 이용하여 빠르고 공정하게 비교 하는 것이 가능하다.

제안된 S^2 알고리즘의 성능 평가를 위해서는 Control Flow에 기반한 프로그램 소스 코드의 유사도를 비교하는 실험을 수행하였고, 그 결과 DP기반의 기존 알고리즘을 사용하였을 때에 비하여 행렬의 크기에 영향을 받지 않고, 정당하게 유사도를 측정하였고, 그 결과가 클러스터링에 반영된 것을 확인하였다.

이 후로는 S^2 알고리즘을 통해 효율적인 생물학적 정보의 정렬에 관해 연구를 진행할 계획이다.

참 고 문 헌

[1] W. Shouzhi, W. Xuemou, "Generalized Principle of Optimality in Pansystems Network Analysis," Kybernetes, Vol. 13(4), pp. 231-235, 1984.

[2] S. Needleman, C. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," Journal of Molecular Biology, Vol. 48(3), pp. 443 - 453, 1970.

[3] M. Gribskov, A. D. McLachlan, and D. Eisenberg, "Profile Analysis: Detection of Distantly Related Proteins," Proc Natl Acad Sci U S A. 84(13), pp. 4355 - 4358, Jul., 1987.

[4] 강은미, "유전체 서열의 정렬 기법을 이용한 소스 코드 표절 검사," 한국정보과학회논문지 Vol. 9(3), pp. 352 - 367, Jun., 2003.

[5] M. A. Trick, "A Tutorial on Dynamic Programming," CMU, 1997.

[6] G. J. Barton, "General Purpose Cluster Analysis Program," Univ. of Dundee, 2002.