제26회 한국정보처리학회 추계학술발표대회 논문집 제13권 제2호 (2006. 11)

생물학적 데이터 서열들에서 빈번한 최대길이 연속 서열 마이닝

강태호*, 유재수**
*충북대학교 정보통신공학과
**충북대학교 전기전자컴퓨터공학부
e-mail: thkang@netdb.cbnu.ac.kr

Mining Maximal Frequent Contiguous Sequences in Biological Data Sequences

Tae-Ho Kang*, Jae-Soo Yoo**
*Dept of Computer and Communication Engineering, Chungbuk National University
**School of Electrical and Computer Engineering, Chungbuk National University

요 약

생물학적 데이터 서열에는 크게 DNA 서열과 단백질 서열이 있다. 이들 서열 데이터들은 여러 데이터베이스에 걸쳐 매우 방대한 양을 가지고 있으며, 각각의 서열은 수백 또는 수천 개의 항목들을 가지고 있어 길이가 매우 길다. 일반적으로 유전적인 변형, 또는 변이로부터 보존된 영역이나 특정 패턴들을 서열 안에 포함하고 있는데 생물학적 서열 데이터에서 보존된 영역이나 패턴들은 계통발생학적 근거로 활용 될 수도 있으며 기능과 밀접한 관계를 가지기도 한다. 따라서 서열들로부터 빈번하게 발생하는 패턴을 발견하고자 하는 알고리즘 개발이 요구되고 있다. 초창기 Apriori 알고리즘을 변형하여 빈발 패턴을 발견하고자 하는 노력들로부터 근래에는 PrefixSpan 트리를 이용하여 효과적으로 성능을 개선하고 있지만 아직까지는 여러 번의 데이터베이스 접근이 요구되고 있어 성능저하가 발생한다. 이에 본 논문에서는 접미사 트리를 변형하여 데이터베이스 접근을 획기적으로 줄이고 많은 서열들로부터 빈번하게 발생하는 연속적인 서열을 효과적으로 발견하는 방법을 제안한다.

1. 서론

최근 생물정보학 분야의 성장에 따라 생물학적 정보의 양은 급속도로 증가하게 되었다. 방대한 생물학적 정보들을 빠르고 효과적으로 분석하기 위한 생물정보학기술들이 요구되고 있다. 그중 생물학적 데이터들의 서열에 대한 상동성 비교 및 분석 방법들은 대량의 생물학적 데이터 분석 해야하는 대표적인 생물정보학 기술이라 할 수 있다.

생물학적 서열 데이터는 크게 DNA 염기 서열과 단백질 아미노산 서열이 있다. 이들 서열 데이터들은 일반적으로 여러 데이터베이스에 걸쳐 매우 방대한 양을 가지고 있으며, 각각의 서열은 수백 또는 수천 개의 항목들을 가지고 있어 그 길이가 매우 길다. 일반적으로 유전적인 변형, 또는 변이로부터 보존된 영역이나 특정 패턴들을 서열 안에 포함하고 있는데 생물학적 서열 데이터에서 보존된 영역이나 패턴들은 계통발생학적 근거로 활용 될 수도 있으며 기능과 밀접한 관계를 가지기도 한다.

* 이 논문은 2006년도 교육인적자원부 지방연구중심대학 육성사업의 지원에 의하여 연구되었음

빈번한 최대길이 연속 서열 마이닝은 두 개 이상의 서열들로부터 빈번하게 발생하는 최대 길이의 연속 패턴을 발견하고자 하는 것이다[1][2]. 이러한 빈번하게 발생하는 최대 길이의 연속 패턴을 찾는 문제는 생물정보학 분야에서 매우 중요한 문제이며 널리 사용된다[3]. 두 개 이상의 DNA 염기 서열이 주어졌을 때 이들의 생물학적 상관관계를 파악하기 위해 빈번하게 발생하는 최대길이 부분 연속 서열을 사용할 수 있다. 이를 위해 초창기 Apriori 알고리즘 기반의 알고리즘을 변형하여 이들 빈발 연속 패턴을 발견하고자 하는 노력들[4][5]로부터 근래에는 PfefixSpan을 이용하여 성능을 개선하고 있다[6][7]. 하지만 여전히 두 개 이상의 서열들로부터 빈발하게 발생하는 최대길이의 부분 연속 서열을 찾기 위한 데이터베이스 접근에 많은 비용이 소요되거나 별도의 데이터 유지공간을 필요로 한다.

본 논문에서는 별도의 데이터 유지 공간을 줄이고 데이

터베이스 접근 횟수를 줄이고자 부분패턴 발견에 접미사 트리를 사용한다[9]. 기본적으로 접미사 트리는 텍스트의 모든 접미사에 대해 만들어진 트라이 데이터 구조이기 때문에 전체 텍스트에 대한 접미사 트리를 구축하기에는 구축 비용이 많이 든다. 따라서 본 논문에서는 트리의 깊이를 미리 지정한 값으로 고정시켜 접미사 트리를 구축한다. 그리고 구축된 접미사 트리를 통해 효과적으로 빈번하게 발생하는 최대길이 부분 연속 서열의 후보 집합을 생성한다.

본 논문의 구성의 다음과 같다. 먼저 2장에서 최근까지의 관련연구를 설명하고 문제점을 제시한다. 그리고 3장에서 문제점을 해결하기위해 새롭게 제안하는 알고리즘을 설명하고 4장에서 제안하는 알고리즘의 특징을 분석한다. 마지막으로 5장에서 결론을 맺는다.

2. 관련연구

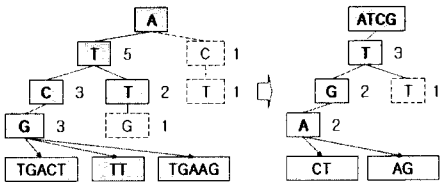
두 개 이상의 서열들로부터 빈발하게 발생하는 최대길이의 부분 연속 서열을 찾기 위해 많은 연구들이 수행되어져 왔다. 이전에 제안된 대부분은 알고리즘들은 빈발하지 않은 서열 패턴들로 이루어진 수퍼 집단들은 빈발하지 않다는 Apriori 기반의 변형된 알고리즘들이었다[4][5]. 대표적으로 GSP 알고리즘의 경우 순차 패턴을 찾기 위해 여러 가지의 경로에 대한 후보 집단을 생성하고 이를 테스트하는 형태로 순차 패턴을 찾는데, 이는 서열의 길이가 커지면 검색 시간이 기하급수적으로 증가한다는 문제가 있다. 이를 개선하는 Apriori 기반의 알고리즘으로 최근에 PrefixSpan 알고리즘이 제안되었다[6][7]. PrefisSpan 알고리즘은 길이가 1인 빈발한 각 항목으로부터 시작하는 프로젝션 데이터베이스를 생성하여 순차 패턴을 확장시켜 나가면서 빈발한 최대길이의 순차패턴을 찾는다. 하지만 순차패턴을 확장할 때 마다 반복적으로 데이터베이스에 접근해 프로젝션 데이터베이스를 생성하여야 되기 때문에 이 알고리즘은 DNA 염기서열이나, 단백질 아미노산 서열과 같이 길이가 긴 서열 데이터에 대해서는 비효율적이다. 마지막으로 각 데이터 항목에 대해 순차적으로 이어지는 서브 시퀀스들로 이루어진 프로젝션 데이터베이스를 각 항목 데이터에 대해 한 번씩 생성하고 이를 고정 길이의 span method를 이용하여 각 데이터 항목에 대한 최대길이 서브 시퀀스를 구해서 이들을 접미사 트리를 이용해 최종적으로 빈발하게 발생하는 최대길이의 부분 연속 서열을 찾는 알고리즘이 있다[7]. 이 알고리즘은 다음과 같다. 먼저 다음의 표 1과 같은 DNA 서열 데이터베이스가 있다.

<표 1>DNA 서열 데이터베이스

ID	sequence
10	ATCGTGACT
20	CATCGTT
30	CATCGTGAAG
40	TCGTGATTG
50	GCGTGATT

표 1의 데이터베이스에서 전체 서열들에서 2번 이상 발생하는 길이 7의 최대 연속 서열이 존재함을 알 수 있다.

이를 찾기 위해 먼저 각 항목 데이터(A, C, T, G)들 각각에 대한 프로젝션 데이터베이스를 작성한다. A 항목에 대한 프로젝션 데이터베이스는 : <TCGTGACT>, <CT>, <TCGTT>, <TCGTGAAG>, <AG>, <G>, <TTG>, <TT>이다. 이에 대한 고정 길이의 span method를 이용하여 A로 시작하는 최소 지지도 2를 만족하는 최대길이 서열을 구하는 과정은 다음과 같다.



(그림 1) 고정길이 span method

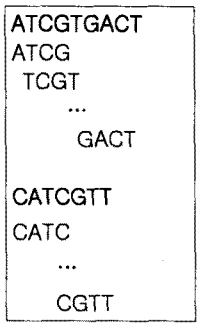
A의 프로젝션 데이터베이스를 고정길이가 3인 span method로 구성하는 것은 그림 1과 같다. 이때 비 단말 노드는 중복되는 접두사를 나타내고 단말노드는 서브 시퀀스를 나타낸다. 서브 시퀀스는 그림 1의 오른쪽과 같이 ATCG를 루트로 하는 고정 길이 span method가 다시 적용된다. 결과적으로 이 과정을 통해 나온 A로 시작하는 최대길이 연속 서열은 <ATCGTGA> 이다. 계속해서 C, T, G 에 대해서도 같은 과정을 반복한다. 마지막으로 각 항목에 대한 최대 길이 서열들에 대해서 접미사 트리를 생성함으로서 최종적인 최대길이 서열을 찾을 수 있다. 이 알고리즘은 데이터베이스 접근을 효율적으로 줄이면서 빈발한 최대길이 연속 서열을 찾을 수 있게 한다. 하지만 각 항목에 대한 프로젝션 데이터베이스들은 그 크기가 매우 크다는 사실을 위의 예에서 알 수 있다. 또한 항목의 종류가 많을수록 즉, 20가지 항목으로 이루어져 있는 아미노산 서열과 같은 경우 프로젝션 데이터베이스를 각 항목별로 구하는 것은 매우 큰 비용을 차지하며 비효율적이다.

3. 제안하는 알고리즘

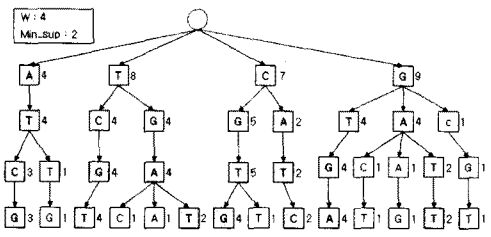
본 논문에서는 프로젝션 데이터베이스를 생성하지 않고 효과적으로 빈발한 최대길이 연속 서열을 찾을 수 있는 알고리즘을 제시한다.

알고리즘은 크게 2가지의 과정으로 나뉜다. 첫 번째, 데이터베이스를 스캔하면서 고정길이(w)의 접미사 트리를 구축한다. 두 번째, 접미사 트리를 통해 최소 지지도를 만족하는 고정길이(w)의 연속 서열 집합을 구하고 이들 집합들을 비교하면서 연속 서열의 길이를 확장시켜나간다. 각 과정은 다음에서 자세하게 설명한다.

먼저 표1의 서열 데이터베이스를 데이터 고정길이를 4로하고 고정길이 만큼 읽으면서 길이 4의 접미사 트리를 구성한다. 읽는 방법은 다음과 같다.



전체 서열을 위와 같은 방식으로 고정길이 4씩 읽으며 접미사 트리를 구축하면 그림 2와 같다. 트리의 각 노드는 카운트를 포함하고 있어 부분 서열의 중복 횟수를 유지한다.



(그림 2) 고정길이 접미사 트리 구축

그림 2는 전체 서열 데이터베이스의 각 서열 데이터들로부터 발생하는 길이가 4인 모든 부분 서열들에 대한 발생 횟수 정보를 포함하고 있는 접미사 트리이다.

이처럼 접미사 트리가 구축되면 곧바로 트리를 순회 하면서 카운터 검색을 통해 최소 지지도를 만족하면서 길이가 4인 연속 서열을 구할 수 있다. 이렇게 구해진 연속 서열은 <ATCG>,<TCGT>,<TGAT>,<CGTG>,<CATC>, <GTGA>, <GATT> 이다. 이들 데이터는 비교하고 있는 모든 서열들에서 최소 지지도 이상으로 발생하는 길이가 4인 모든 부분 서열이면서 또한 최대 길이 연속 서열을 이룰 가능성이 있는 후보 집단이 된다. 즉, 다시 말해 최소 지지도를 만족하는 최대 길이 연속 서열은 위의 서열들로 이루어진다. 이것은 Apriori 알고리즘에서의 빈발하지 않은 서열 패턴들로 이루어진 수퍼 집단들은 빈발하지 않다는 사실에 기인한다.

다음으로 이들 길이가 4인 빈발 연속 서열들에서 서로 이어질 가능성이 있는 서열들을 묶어 서열을 확장시키면서 최대 길이 연속 서열의 후보 집단을 생성한다. 서열을 묶을 때는 <ATCG>와 <TCGT>의 경우에서와 같이 <ATCG>의 처음 A를 제외한 나머지부분과 <TCGT>의 마지막 T를 제외한 나머지 부분이 일치하는 경우 <ATCGT>와 같이 연결되어 확장될 수 있다. 그 결과 <ATCGT>, <TCGTG>, <TGATT>, <CGTGA>, <CATCG>, <GTGAT> 가 되고 이것을 다시 같은 방법으로 더 이상의 확장이 불가능 할 때까지 반복적으로 확장한다. 이 과정을 통해 최종적으로 얻어지는 최대길이 항목 집단의 전체 후보 집합은 다음과 같다.

<표 2> 최대길이 연속 서열 후보 집단 산출

길이 4	길이 5	길이 6	길이 7
ATCG	ATCGT	ATCGTG	
TCGT	TCGTG	TCGTGA	CGTGATT
TGAT	TGATT	CATCGT	CATCGTG
CGTG	CGTGA	CGTGAT	ATCGTGA
CATC	CATCG	GTGATT	
GTGA	GTGAT		
GATT			

표 2의 결과는 길이가 4인 빈발한 부분 연속 서열로부터 최대 확장 가능한 서열들을 예측한 것이다. 따라서 실제의 최대 길이 연속 서열은 후보 집단의 최대 길이로부터 직접 테스트를 통해 구한다. 후보 집단 중에서 최대 길이로부터 테스트를 하여 결과가 도출되면 그보다 길이가 작은 후보 집단은 테스트할 필요가 없다. 결과적으로 표 1의 서열 데이터베이스에서 실제로 빈발하게 발생한 최대길이 연속 서열은 <ATCGTGA> 와 <CGTGATT>인 것을 확인할 수 있다. 이 알고리즘은 최소 지지도 3 또는 4와 같이 다양한 최소 지지도에 대한 최대길이 연속 서열을 주어진 접미사 트리를 이용하여 효과적으로 적용할 수 있다. 특히 높은 최소 지지도에 대해서는, 예를 들어 최소 지지도 5의

경우 부분 후보 집단을 생성하지 않고 트리검색만으로 최대 길이 연속 서열을 확인할 수 있다.

4. 제안하는 알고리즘 분석

본 논문에서 제안한 알고리즘은 이전 연구들에 비해 다음과 같은 특징을 가진다. 첫 번째, 한 번의 데이터베이스 접근과 접미사 트리를 구축으로 여러 값의 최소 지지도를 효과적으로 수용할 수 있다. 두 번째, 접미사 트리를 구성하는 고정길이를 크게 할수록 상대적으로 검색 성능을 높일 수 있다. 즉, 시스템 성능과 검색 성능을 적절히 고려하여 유연하게 대처할 수 있다. 세 번째, 고정길이에 따른 성능 향상과 더불어 최소 지지도가 높을 경우 부분 후보 집단 생성 없이 트리 검색만으로도 결과 도출이 가능하다. 네 번째, 길이가 긴 서열일수록 다른 알고리즘에 비해 높은 성능을 보인다. 마지막으로 항목(차원) 수가 적은 DNA 서열뿐만 아니라 항목 수가 많은 단백질 아미노산 서열 및 기타 다차원의 서열데이터에 대해서도 적용 가능하다.

5. 결론

본 논문에서는 생물정보학 분야에서 매우 중요하게 다뤄지고 있는 빈발한 최대 길이 연속 서열 탐색 문제를 매우 빠르고 효과적으로 처리할 수 있는 알고리즘을 제안하였다. 기존 알고리즘에 비해 데이터베이스 접근 횟수를 획기적으로 줄이고 이전 알고리즘들에서 필요로 하던 부가적인 데이터 산출과정을 없앰으로서 서열데이터에 대한 비교 및 검색 성능을 높일 수 있었다. 현재는 이 알고리즘을 최적화하기 위해 트리의 레벨, 다양한 최소 지지도, 다양한 길이의 서열데이터, 다차원의 서열데이터 등의 다양한 환경을 적용하면서 성능평가를 수행하고 있다. 향후, 이러한 성능평가 결과를 기반으로 알고리즘을 수정 및 보완하여 더욱 완성도 높은 논문으로 만들어갈 계획이다.

참고문헌

[1] V. Chvatal and D. Sankoff "Longest Common Subsequences of two random Sequences" Applied Probability, 12, 306-315, 1995.

[2] R. Wanger and M. Fischer "The string-to-string Correction Problem" ACM, 21, 168-173, 1974.

[3] S. Needleman, C. Wunsch "A general Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins" Mol. BioInformatics, 48(3), 443-453, 1970.

[4] R. Agrawal and R. Srikant "Fast algorithms for mining association rules" In Proc. 1994 int. Conf. VeryLarge DataBases(VLDB'94), 487-499, Santiago, Chile Sept. 1994.

[5] R. Srikant and R. Agrwal "Mining Sequential Patternrs: Generalizations and performance improvements" In proc. 5th Int. Conf. Extending Database Technology(EDBT'96), 3-17, Avignon, France, Mar. 1996.

[6] J. pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth" In ICDE'01, Gemany, April 2001.

[7] Jin Pan, Peng Wang Wei Wang Baile Shi and Genxing Yang "Efficient Algotithms for Mining Maximal Frequent Concatenate Sequences in Biological Datasets" Datamining and Knowledge Discovery 87-116 2005.

[8] D. Hirschberg "Algorithms for the logest common subsequence problem" the Assoc. Comput. Mach, 24(4), 664-675, 1997

[9] E.M. McCreight "A space-economical suffix Tree construction algorithms" ACM 23 262-272 1976.

[10] M. farach "Optimal suffix tree construction with large alphabets" IEEE Symp. Found Computer Science 137-143, 1997.

[11] R. Hariharan "Optimal parallel suffix tree construction" IEEE Symp. Found Computer Science, 290-299, 1994.