DOI: 10.3745/KIPSTD.2008.15-D.2.155

생물학적 데이터 서열들에서 빈번한 최대길이 연속 서열 마이닝

강 태 호[†] · 유 재 수[††]

요 약

DNA 염기 서열이나 단백질 아미노산 서열과 같은 생물학적 서열 데이터들은 일반적으로 많은 수의 항목들을 가지고 있다. 생물학적 데이터 서열들에는 보통 빈번하게 발생하는 수 백개의 항목으로 이루어진 연속된 서열들이 존재한다. 이들 서열들에서 빈번하게 발생하는 연속 서열을 검색하는 것은 생물학적 서열 분석에서 중요한 부분을 차지하고 있다. 이전에는 순차 패턴을 효과적으로 발견하고자 하는 많은 연구들이 수행되었으며 대부분의 기존 순차패턴 마이닝 기법들은 Apriori 알고리즘을 기반으로 한다. PrefixSpan 알고리즘은 Apriori 기반의 가장 효율적인 순차패턴 마이닝 기법이다. 하지만 이 알고리즘은 길이-1인 빈발 패턴들로 부터 서열 패턴을 확장해나가는 방식이다. 따라서 길이가 긴 연속 서열을 포함하는 생물학적 데이터서열들에 대한 검색방법으로는 적합하지 않다. 최근에는 기존의 PrefixSpan방식을 이용하면서도 반복적인 처리과정을 줄인 MacosVSpan이 제안되었다. 하지만 이 알고리즘 또한 길이가 긴 생물학적 데이터 서열들로부터 빈번하게 발생하는 연속 서열들을 검색하기에는 효율적이지 않다. 본 논문에서는 많은 양의 생물학적 데이터 서열들로부터 빈번한 연속서열을 고정길이 확장 트리를 이용하여 효과적으로 찾아내는 방법을 제안한다. 그리고 다양한 환경에서 실험을 통해 제안하는 방식이 MacosVSpan알고리즘에 비해 검색성능이 보다 우수함을 보인다.

키워드 : 바이오인포매틱스, 시퀀스 패턴 마이닝, 보존서열 추출, 서열비교

Mining Maximal Frequent Contiguous Sequences in Biological Data Sequences

Tae-Ho Kang[†] · Jae-Soo Yoo[††]

ABSTRACT

Biological sequences such as DNA sequences and amino acid sequences typically contain a large number of items. They have contiguous sequences that ordinarily consist of hundreds of frequent items. In biological sequences analysis(BSA), a frequent contiguous sequence search is one of the most important operations. Many studies have been done for mining sequential patterns efficiently. Most of the existing methods for mining sequential patterns are based on the Apriori algorithm. In particular, the prefixSpan algorithm is one of the most efficient sequential pattern mining schemes based on the Apriori algorithm. However, since the algorithm expands the sequential patterns from frequent patterns with length-1, it is not suitable for biological dataset with long frequent contiguous sequences. In recent years, the MacosVSpan algorithm was proposed based on the idea of the prefixSpan algorithm to significantly reduce its recursive process. However, the algorithm is still inefficient for mining frequent contiguous sequences from long biological data sequences. In this paper, we propose an efficient method to mine maximal frequent contiguous sequences in large biological data sequences by constructing the spanning tree with the fixed length. To verify the superiority of the proposed method, we perform experiments in various environments. As the result, the experiments show that the proposed method is much more efficient than MacosVSpan in terms of retrieval performance.

Key Words : Bioinfomatics, Sequence Pattern Mining, Conserved Region Extraction, Sequence Comparison

1. 서 론

최근 생물정보학 분야의 성장에 따라 생물학적 정보의 양은 급속도로 증가하게 되었고 방대한 생물학적 정보들을 보다 빠르고 효과적으로 분석하기 위한 생물정보학 기술들이 요구되고 있다. 생물정보학에 의한 유전자 기능분석 및 예측은 기존의 생물학적 기법에서 요구되었던 엄청난 비용을 절감하고 또한 실험적 검증에 소요되는 많은 시간을 단축시켜준다. 유전자 기능 분석에는 전사관련 신호부위 및 조절

※ 이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임(지방연구중심대학육성사업/충북BIT연구중심대학육성사업단)
† 준 회 원 : 충북대학교 전기전자컴퓨터공학부 Post-Doc
†† 종신회원 : 충북대학교 전기전자컴퓨터공학부 정교수
논문접수 : 2007년 3월 21일, 심사완료 : 2008년 1월 22일

인자 결합영역 예측, 프로모터 영역 예측, 단백질상의 특정 기능부위 예측, 핵산 및 단백질 서열 상동성 비교 등이 있으며 이러한 분석의 결과는 신규유전자 발굴 및 기능분석에 중요한 실마리를 제공한다. 이 중에서도 진화론적으로 관련 있는 단백질이나 DNA 서열들이 기능 및 구조적으로 공유하는 공통의 특징에 대한 생물학적 실체인 모티프(Motif) 및 도메인(Domain)에 대한 검색은 유전체의 기능 분석과 분류에서 매우 중요한 역할을 한다.

생물학적 데이터 서열들에 대한 상동성비교 및 서열 분석 방법들은 대량의 생물학적 데이터 분석을 위한 대표적인 생물정보학 기술이라 할 수 있다. DNA 서열 결정이 급속히 진전되면서 같은 그룹의 여러 개의 핵산이나 단백질 서열에 대한 비교가 필요하게 되고, 단백질 패밀리나 관련 핵산의 모티프(motif) 서열 등에 대한 자료가 증가하고 기능 분석 연구가 활발해지고 있다. 최근 들어 서열이 결정되고 방대한 양의 유전자 관련 실험 결과들이 도출됨에 따라 이러한 생물정보학적 기법에 의한 분석 결과들은 더욱 중요하게 활용되고 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. 이러한 생물학적 서열들은 일반적으로 유전적인 변형 또는 변이로부터 보존된 영역(Conserved Region)이나 특정 서열 패턴(Sequence Pattern)들을 서열 안에 포함하고 있는데 여러 개의 생물학적 서열 데이터들에 공통적으로 존재 하는 보존된 영역이나 특정 패턴들은 계통발생학적(Phylogenetic) 근거로 활용 될 수도 있으며 기능과 밀접한 관계를 가지기도 한다.

빈번한 최대길이 연속 서열 마이닝 기법은 여러 개의 서열들에서 존재하는 서브시퀀스 패턴을 보다 효율적으로 추출할 수 있는 알고리즘이다. 이 알고리즘은 또한 공통적으로 보존된 서브 시퀀스들을 추출함으로서 두 개의 서열에 대한 상동성을 비교하는데 활용되거나 여러 개의 부분 서열들로부터 서로 일치하는 서열들을 찾아 하나의 긴 서열로 병합하는 응용 등에 활용될 수 있다[1][2][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 알고리즘을 변형하여 이들 빈발 연속 패턴을 발견하고자 하는 노력들[4][5]로부터 근래에는 프로젝션 데이터베이스와 PrefixSpan 트리를 이용한 방법으로 성능 개선을 위한 노력들이 시도되고 있다[6][7]. 하지만 두 개 이상의 서열들로부터 빈번하게 발생하는 최대길이의 부분 연속 서열을 찾기 위해서는 여전히 여러 번의 데이터베이스 접근이 요구되거나 별도의 프로젝션 데이터베이스 생성을 요구하고 있어 많은 비용이 소요되거나 많은 양의 데이터 유지공간이 별도로 필요한 문제점이 있다.

따라서 본 논문에서는 불필요한 데이터를 없애 별도의 데이터 유지 공간을 줄이고 데이터베이스 접근 횟수를 획기적으로 줄이면서 여러 서열 데이터로부터 빈번하게 발생하는 부분 서열패턴을 효율적으로 추출할 수 있는 알고리즘을 제안한다. 본 논문의 구성의 다음과 같다. 먼저 2장에서 최근까지의 관련연구를 설명하고 문제점을 제시한다. 그리고 3장에서 문제점을 해결하기위해 새롭게 제안하는 알고리즘을 설명하고 제안하는 알고리즘의 특징을 분석한다. 4장에서는 기존에 제안된 알고리즘들과의 성능평가를 통해 본 알고리즘의 우수성을 설명하고 마지막으로 5장에서 결론을 맺는다.

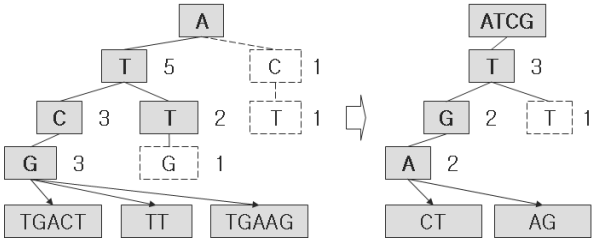
2. 관련연구

두 개 이상의 서열들로부터 빈번하게 발생하는 최대길이의 부분 연속 서열을 찾기 위해 많은 연구들이 수행되어왔다. 제안된 대부분의 기존 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 가지 경로에 대한 후보 집단을 생성하고 이를 테스트하는 형태로 순차 패턴을 찾는데, Apriori 기반 알고리즘의 특성상 서열의 길이가 커지면 그 만큼 데이터베이스 접근 횟수가 늘기 때문에 검색시간이 기하급수적으로 증가한다는 문제가 있다. 이를 개선하는 Apriori 기반의 알고리즘으로 최근에 PrefixSpan 알고리즘이 제안되었다[6][7]. PrefixSpan 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 빈번하게 발생하는 최대길이의 부분 연속 서열을 찾는 MocosVSpan알고리즘이 있다[7]. 알고리즘 설명을 위해 먼저 표 1과 같은 DNA 서열 데이터베이스가 있다고 가정한다.

<표 1>의 DNA 서열 데이터베이스에 있는 전체 서열들에

〈표 1〉DNA 서열 데이터베이스

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



(그림 1) 고정길이 span method

서 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. (그림 1)은 A-프로젝션 데이터베이스로부터 span method를 이용하여 A로 시작하는 최소 지지도 2를 만족하는 최대길이 서열을 구하기 위한 고정길이가 3인 확장 트리를 보인다.

트리에서의 비 단말 노드는 중복되는 접두사를 나타내고 단말노드는 서브 시퀀스를 나타낸다. 단말노드의 서브 시퀀스들은 (그림 1)의 오른쪽과 같이 지지도를 만족하는 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 최대길이 서열들에 대해서 접미사 트리를 생성함으로서 최종적인 최대길이 서열을 찾을 수 있다.

이 알고리즘은 PrefixSpan 방식을 사용하면 서도 서열 패턴을 확장하기 위해 반복적인 수행과정을 많이 감소 시켰다. 하지만 MacosVSpan 알고리즘의 경우도 프로젝션 데이터베이스를 별도로 생성하고 이를 이용한다는 문제점이 있다. 프로젝션 데이터베이스는 원본 데이터베이스에 비해 상당히 큰 데이터양을 차지하는데 데이터 서열의 길이가 길어질수록 프로젝션 데이터베이스의 양은 기하급수적으로 늘어난다는 문제점을 가지고 있다. 이에 대한 자세한 설명은 4.1절의 프로젝션 데이터베이스의 문제점에서 다룬다. 프로젝션 데이터베이스를 이용한다는 것은 각 항목별 프로젝션 데이터베이스들은 4.1절의 실제 데이터 비교를 통해 그 크기가 매우 크다는 사실을 알 수 있고 또한 항목의 종류가 많을수록 즉, 20가지 항목으로 이루어져 있는 아미노산 서열과 같은 경우 프로젝션 데이터베이스를 각 항목별로 구하는 것은 매우 큰 비용을 차지하며 비효율적임을 알 수 있다.

3. 제안하는 최대길이 연속 서열 마이닝 알고리즘

별도의 프로젝션 데이터베이스를 생성하지 않고 여러 개의 생물학적 데이터 서열들로부터 효과적으로 빈발한 최대길이 연속 서열을 찾을 수 있는 알고리즘을 제시한다.

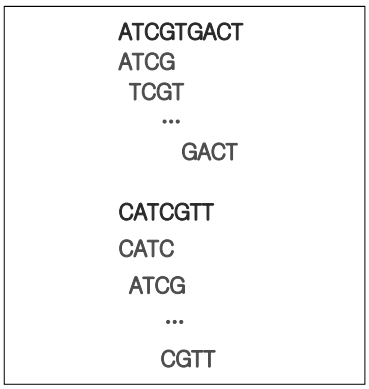
```
Algorithm
Input : Sequences Set S(S1, S2.. SN), FixedLength W, Minimum
Support threshold Min_Sup
Output : fixed length suffix trie T, MFCSeq

Step 1: 데이터베이스로부터 고정길이 서열 추출 및 인덱스 구축
1. for(i=0; i< N; i++)
2. WS = extractFixedLengthSubseq(W, Si); //부분 시퀀스 추출
3. memTree = constructTree(W); // 트리 구축
4. for(i=0; i< WS; I++)
5. insertSequences(WSi, memTree); //시퀀스 삽입 및 카운트

Step 2: 인덱스 검색 및 후보집합 산출
6. candidateSeq = searchTree(memTree, Min_sup); //후보서열
   추출
7. resultSeq = makeSeqCandidate(candidateSeq); // 후보서열
   확장

Step 3: 후보 서열과 데이터베이스 비교검색
8. MFCSeq = searchDatabase(resultSeq); //최대길이 서열 비교
```

(그림 2) 최대길이 연속서열 마이닝 알고리즘

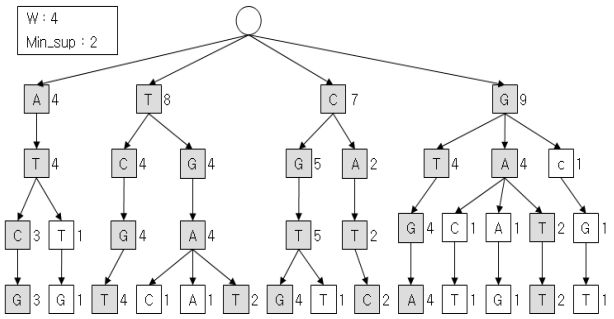
(그림 2)는 본 논문에서 제안하는 최대길이 연속 서열 알고리즘을 보여준다. 내용을 요약하면 알고리즘은 크게 3가지 단계로 나뉜다. 먼저 첫 번째 단계에서는 주어진 서열들로부터 고정길이 윈도우 길이의 부분 서열들을 읽어서 확장 트리를 구축한다. 두 번째 단계에서는 구축된 트리를 깊이 우선 탐색을 통해 지지도를 만족하는 부분서열들을 산출하고 이들로부터 확장하여 전체 후보 서열을 산출한다. 세 번째 단계에서는 최종적으로 산출된 후보 서열과 데이터베이스를 비교하여 실제 지지도를 만족하는 가장 길이가 긴 서열을 추출한다. 이때 비교는 길이가 긴 후보 서열들로부터 길이가 짧은 서열 순서로 비교하며 만족하는 결과를 얻었을 때 비교검색을 중단 한다. 알고리즘 수행에 관한 보다 자세한 설명은 예제를 통해 설명한다.

(그림 2)의 (그림 3)번 라인에서는 트리 구축을 위한 자료구조를 정의하고 루트 노드를 생성한다. 다음으로 5번 라인에서는 라인 2에서 추출된 전체 고정길이 서열들을 반복적으로 확장트리에 삽입하고 이들의 발생횟수 카운트를 기재한다. 그리고 6번 라인에서 구축된 트리를 깊이 우선방식으로 순회하며 리프 노드(leaf node)의 카운트가 최소 지지도 이상인 경우 최상위 노드로부터 리프 노드까지의 서열을 후보 서열로 저장한다. 라인 7에서는 추출된 후보서열들을 확장 가능성을 판단하여 길이 1씩 증가시키며 서열을 확장하는 과정을 반복적으로 수행하여 확장 가능한 최대길이까지 확장시킨다. 그리고 마지막으로 라인 8에서 원본 데이터베이스에 대한 순차 스캔을 통해 후보 서열들에 대한 실재 존재 여부를 확인한다.

알고리즘 설명을 위해 <표 1>과 같은 데이터베이스가 있다고 가정한다. 그리고 고정길이를 4로 최소 지지도를 2로



(그림 3) 고정길이 스캔 방법



(그림 4) 고정길이 확장 트리 구축

가정한다. 먼저 표1의 서열 데이터베이스를 데이터 고정길이 윈도우의 크기를 4로 하여 처음 위치로부터 하나씩 이동하면서 고정길이 윈도우만큼 읽어 길이 4의 고정길이 확장 트리를 구성한다. (그림 3)는 서열 데이터를 고정길이 만큼 읽는 것을 보이고 있다. 전체 서열을 (그림 3)과 같은 방식으로 지정된 고정길이 만큼 읽으며 확장 트리를 구축하면 (그림 4)와 같다. 트리의 각 노드는 카운트를 포함하고 있어 부분 서열의 중복 횟수를 유지한다.

(그림 4)는 전체 서열 데이터베이스의 각 서열 데이터들로부터 발생하는 길이가 4인 모든 부분 서열들에 대한 발생 횟수 정보를 포함하고 있는 확장 트리이다. 이처럼 고정길이 확장 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 때

〈표 2〉 최대길이 연속 서열 후보 집단 산출

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

〈표 3〉 서열들에 존재하는 실제 부분 연속 서열

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

는 <ATCG>와 <TCGT>의 경우에서와 같이 <ATCG>의 처음 A를 제외한 나머지 부분과 <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 <표 2>와 같다. <표 2>는 길이가 4인 부분 연속 서열로부터 최대 확장 가능한 서열들을 예측한 것이다.

<표 3>은 서열들에서 일정 지지도를 만족하면서 실제로 존재하는 서열들을 검사한 것이다. <표 2>에서 지지도를 만족하는 길이 4의 부분 서열들로부터 점차로 확장해 나간 후보 서열과 <표 3>에서 보이는 실제 지지도를 만족하는 부분 서열과의 차이가 크지 않음을 확인할 수 있다. 따라서 확장을 통한 후보 부분 서열이 도출된 결과가 실제 존재하는 부분 서열과 매우 유사함을 알 수 있고 이를 통해 최대길이의 후보 서열 또한 유추가 가능함을 알 수 있다. 그리고 이러한 후보서열 도출 방식의 정확성이 상당히 높음을 <표 2>와 <표 3>의 비교를 통해 확인할 수 있다.

최종적으로 도출된 후보 집단은 최대 길이 후보 서열부터 실제 데이터베이스 검색을 통해 확인하게 된다. 이때 결과가 도출되면 그보다 길이가 작은 후보 집단은 테스트할 필요가 없다. 결과적으로 <표 1>의 서열 데이터베이스에서 실제로 빈번하게 발생한 최대길이 연속 서열은 <ATCGTGA>와 <CGTGATT>인 것을 확인할 수 있다.

MacosVSpan 알고리즘에서의 프로젝션 데이터베이스 대

한 재귀적인 방법은 높은 공간 복잡도를 요구한다. 프로젝션 데이터베이스의 크기를 P라 하고 고정길이를 N, 그리고 반복 횟수를 R이라 할 때 MacosVSpan 알고리즘의 공간복잡도는 초기 프로젝션 데이터베이스 생성비용 P와 고정길이 N만큼의 트리 생성 비용 및 나머지 프로젝션 데이터베이스의 저장 비용이 R만큼 반복적으로 더해져야한다. 하지만 제안하는 알고리즘에서는 고정길이 N만큼의 트리생성 비용만을 요구한다. 그리고 시간 복잡도 측면에서 볼 때 MacosVSpan 알고리즘의 경우 데이터베이스 스캔을 통한 프로젝션 데이터베이스의 생성시간 및 트리 구축시간과 프로젝션 데이터베이스에 대한 재귀적인 트리 구축 및 검색을 요구하지만 제안하는 방식에서는 데이터베이스 스캔을 통한 한 번의 트리구축과 검색 그리고 마지막 한 번의 데이터베이스 검색을 요구하므로 시간복잡도 측면에서도 매우 효율적이라 할 수 있다.

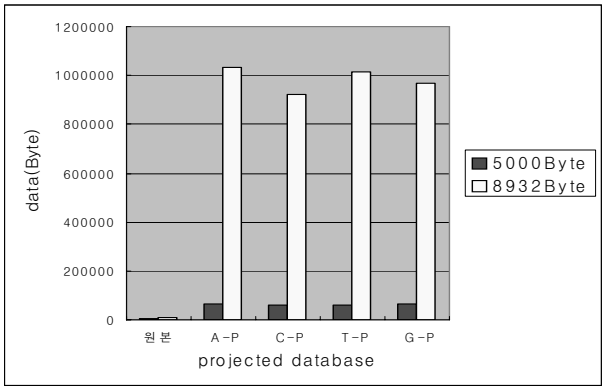
또한 이 알고리즘은 최소 지지도 3 또는 4와 같이 다양한 최소 지지도에 대한 최대길이 연속 서열을 주어진 고정길이 확장트리를 이용하여 효과적으로 적용할 수 있다. 특히 높은 최소 지지도에 대해서는, 예를 들어 최소 지지도 5의 경우 부분 후보 집단을 생성하지 않고 트리검색만으로 최대 길이 연속 서열을 확인할 수 있다.

4. 성능평가

본 장에서는 성능평가를 통해 제안하는 알고리즘의 우수성을 규명한다. 성능평가는 Pentium4 1.8GHz CPU와 512MByte RAM의 하드웨어와 Linux 운영체제 환경에서 C프로그램을 이용하였다. 성능평가에 앞서 이전 연구들에서 여러 개의 서열들에서 공통적으로 존재하는 부분 연속서열 추출을 위해 별도로 생성했던 프로젝션 데이터베이스의 문제점에 대하여 먼저 설명하고자 한다.

4.1 프로젝션 데이터베이스의 문제점

프로젝션 데이터베이스는 일반적으로 원본 서열 데이터베이스에 비해 상대적으로 매우 큰 용량을 차지한다. 이러한 사실을 증명하기 위해 다음과 같은 서열 데이터들을 통해 프로젝션 데이터베이스를 생성하고 이를 비교하였다. <표 4>는 크기가 다른 원본 서열 데이터에 대해서 각각 프로젝션 데이터베이스를 만들고 이에 대한 데이터의 용량을 비교한 표이다. 사용된 데이터는 DNA 서열을 무작위로 생성한 데이터와 실제 DNA 서열 데이터를 사용하였고 각각의 데이터 용량은 5000(Byte), 8932(Byte)이다. 그리고 각 서열들



(그림 5) 프로젝션 데이터베이스의 용량비교

로부터 핵산의 염기 A, C, G, T에 대한 각각의 프로젝션 데이터베이스를 생성한 결과 데이터 용량의 차이를 보이고 있다. 그리고 (그림 5)에서는 <표 4>에서 보이는 차이를 보다 확인하기 쉽게 하기위해 그래프를 이용하여 차이를 비교했다.

<표 4>와 (그림 5)를 통해 원본 서열 데이터의 용량이 커질수록 프로젝션 데이터베이스의 용량이 기하급수적으로 증가함을 알 수 있다. 이에 따라 프로젝션 데이터베이스를 생성하는 비용이 많이 들고 이는 궁극적으로 검색성능 저하에 중대한 영향을 미친다. 그리고 서열의 길이가 길어질수록 프로젝션 데이터베이스의 용량은 더욱 증가하게 된다. 따라서 본 논문에서는 검색성능을 높이고 이러한 문제점을 해결하기 위해 프로젝션 데이터베이스를 생성하지 않고도 빈번하게 발생하는 부분 연속 서열 추출이 가능하도록 하였다.

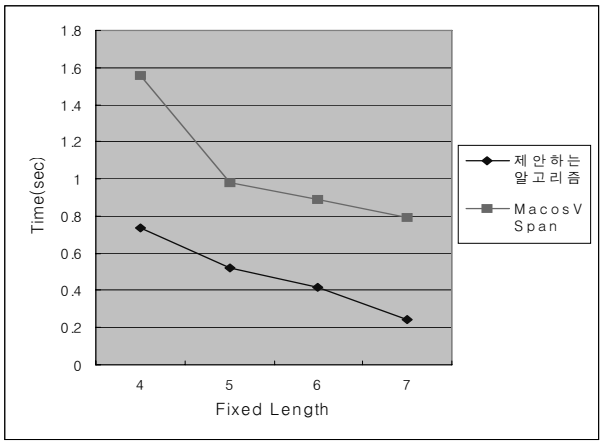
4.2 검색성능 및 메모리 사용량

성능평가를 위한 데이터는 DNA 서열을 이용하였다. 그리고 염기가 고른 분포를 갖는 서열을 얻기 위해 A, C, G, T의 염기로 이루어진 길이가 500인 DNA 서열 데이터 100개를 무작위 추출 방식으로 생성하여 사용하였다.

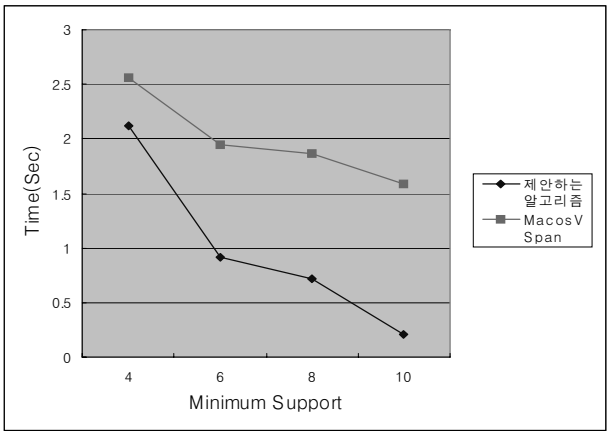
(그림 6)과 (그림 7)은 제안하는 알고리즘과 기존의 MacosVspan 알고리즘의 검색성능을 비교하여 제안하는 알고리즘의 검색성능이 우수함을 보인다. 그림 6, 7에 나타난 검색시간의 결과는 트리를 생성하는 시간과 트리 검색시간을 포함한 총 시간을 나타내며, 본 논문에서는 제시하는 알고리즘에서는 단 한 번의 트리구축과 검색을 필요로 하나 MacosVSpan 알고리즘은 특성상 트리를 여러 번 재구축하고 검색하는 시간을 포함하게 된다. MacosVspan 알

<표 4> 원본 데이터베이스 용량 변화에 따른 프로젝션 데이터베이스의 용량 비교

프로젝션 데이터 / 데이터크기(Byte)	원본 데이터베이스	A-프로젝션 데이터	C-프로젝션 데이터	G-프로젝션 데이터	T-프로젝션 데이터
무작위 생성 서열	5000	66529	61286	64734	59901
실제 DNA 서열	8932	1034951	920142	967499	1015148



(그림 6) MacosVSpan과 제안하는 알고리즘의 검색성능 비교



(그림 7) 최소지지도 변화에 따른 검색속도

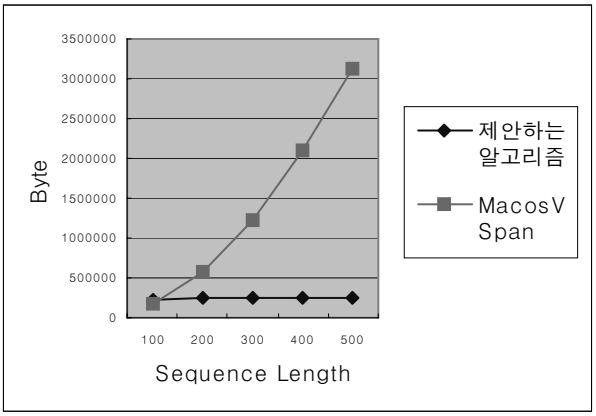
〈표 5〉 고정길이 변화에 따른 메모리 사용량

구분 \ 고정길이	5	6	7	8	9	10
메모리 사용량(Byte)	19112	64168	225692	535892	909892	1296474

고리즘의 경우 길이가 긴 서열 데이터로부터 최대길이의 공통부분 서열을 추출하기 위해서는 관련연구에서 설명하였듯이 각 항목에 프로젝션 데이터베이스를 생성하고 또한 고정길이 확장트리를 여러 번 반복적으로 구축해야하고 이를 위해 프로젝션 데이터베이스를 여러 번 스캔해야 한다. 하지만 제안하는 알고리즘의 경우 단 한 번의 트리구축과 한 번의 데이터베이스 스캔을 필요로 한다. 물론 제안하는 알고리즘에서 예측되는 후보 집합의 경우 실제의 정확한 결과가 아닌 부정확한(False Positive) 데이터를 포함하고 있어 최종적으로 데이터베이스 스캔을 통해 실제 존재 여부를 확인해야 하는 문제점이 있다. 하지만 고정길이를 크게 확장 한다면 도출되는 후보 서열이 적어지기 때문에 부정확한 데이터를 상당수 줄일 수 있게 된다. 이러한 사실은 그림 5의 제안하는 알고리즘에서 고정길이의 변화에 따른 검색성능의 변화를 통해 확인할 수 있는데 고정 길이가 7 이하인 경우의 검색성능보다 7인 경우에서 이전보다 눈에 띄게 성능이 높아짐을 알 수 있다.

다음의 <표 5>는 제안하는 알고리즘에서 고정길이 변화에 따른 메모리 사용량을 나타낸다. 고정길이를 5에서부터 10까지 변화시켜가며 실제 랜덤 데이터를 이용해 트리를 구축해보고 이때 필요한 메모리양을 측정했다. 대략 고정길이를 1 확장할 때마다 필요한 메모리양은 대략 2배정도임을 알 수 있다. 그리고 고정길이 10을 기본으로 했을 때 필요한 메모리양은 1Mbyte 정도로 충분히 수용할 수 있는 크기이며 이는 트리 구현 방식에 따라 줄이는 것도 가능하다.

또한 위와 같이 고정길이를 확장 시키는 것 이외에도 알고리즘의 특성상 검색성능이 높아지는 경우가 있다. 먼저



(그림 8) 두 알고리즘에 사용된 메모리양

추출되는 최대길이 공통부분 서열의 길이가 고정길이보다 작거나 같은 경우는 별도의 확인 작업 없이 곧바로 정확한 결과를 얻을 수 있다. 그리고 빈발 지지도가 높을수록 기존 방식에 비해 후보 데이터가 획기적으로 줄기 때문에 더욱 좋은 검색성능을 보인다.

(그림 7)은 빈발 지지도에 따른 검색결과를 기존방식과 비교한 것이다. 앞에서 설명한 실제 랜덤 데이터를 이용해 고정길이 5인 트리를 구축하였다. 처음 최소 지지도가 4인 부분에서는 기존방식과 검색속도 측면에서 많은 차이가 없음을 확인할 수 있다. 이는 최소 지지도를 만족하는 길이가 5인 짧은 부분 서열이 많이 존재하는 경우로서 제안하는 알고리즘에서도 후보 산출 계산에 필요한 비용이 많아짐을 의미한다. 하지만 최소 지지도가 6일 경우에서는 최소 지지도를 만족하는 길이 5의 부분 서열이 많이 줄어들어 후보 산출 및 전체 검색시간을 많이 감소시키는 결과를 보인다. 지지도 6과 8 사이는 지지도를 만족하는 결과로 검색된 길이 5의 부분서열의 수가 많은 차이를 보이지 않은 경우로 두 지지도 사이의 성능 변화가 크지 않음을 알 수 있다. 마지막으로 지지도가 10인 부분에서는 지지도를 만족하는 길이 5의 서열이 극히 적어 후보 집합 확장 및 산출 없

이 트리 구축 및 검색만으로 최종 결과를 도출하는 경우를 보이고 있다.

(그림 8)은 (그림 6)에서 비교하는 두 방식에서 사용된 메모리 사용량을 보이고 있다. 사용된 데이터는 서열의 길이가 각각 100에서 500까지인 DNA서열 100개를 이용하여 고정길이가 7인 트리를 구축하였다. 그림 8에서 서열의 길이가 100인 비교적 짧은 경우는 기존 방식에 비해 새로 제안하는 알고리즘의 메모리 사용량이 많은데 이는 기존 MascosVspan 알고리즘의 경우 프로젝션 데이터베이스를 하나씩 처리한 후 이들로부터 얻은 결과를 통해 최종 검색 결과를 산출하는 반면에 제안하는 알고리즘은 모든 경우의 확장트리를 한번에 구축하기 때문에 조금 더 많은 메모리 사용량을 요구한다. 하지만 이러한 현상은 비교하는 서열의 길이가 작을 경우에 나타난다. MacosVspan알고리즘은 서열의 길이가 길어질수록 접미어의 크기기 기하급수적으로 커지기 때문에 많은 메모리 용량이 필요하다. 반면 제안하는 알고리즘의 경우 고정길이의 확장 트리는 최대 크기 이후 더 이상 증가하지 않는다. 이는 서열의 길이에 따른 변화에서는 제안하는 알고리즘은 길이가 늘어나더라도 고정길이 스캔을 통해 고정길이만큼의 서열에 대한 빈도만이 증가하기 때문에 고정길이를 표현하기 위해 확보된 메모리 영역 이외에는 더 이상의 메모리를 필요로 하지 않기 때문이다.

4.3 알고리즘 분석

본 논문에서 제안한 알고리즘은 이전 연구들에 비해 다음과 같은 특징을 가진다. 첫 번째, 한 번의 데이터베이스 접근과 고정길이 확장트리를 구축으로 여러 값의 최소 지지도를 효과적으로 수용할 수 있다. 두 번째, 트리를 구성하는 고정길이를 크게 할수록 상대적으로 검색성능을 높일 수 있다. 즉, 시스템 성능과 검색성능을 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 결 론

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 검색성능을 높일 수 있었다. 향후 제안하는 알고리즘을 최적화하기 위해 트리의 레벨, 다양한 최소 지지도, 다양한 길이의 서열 데이터, 다차원의 서열데이터 등의 다양한 환경을 고려하여 최적화할 예정이다.

참 고 문 헌

[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 Patterns: 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 Algorithms for Mining Maximal Frequent Concatenate Sequences in Biological Datasets” Proceedings of the The Fifth International Conference on Computer and Information Technology 98-104, 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.

강 태 호

e-mail : thkang@netdb.cbnu.ac.kr

1999년 호원대학교 정보통신공학과 (공학사)

2002년 충북대학교 대학원 정보산업 공학과 (공학석사)

2007년 충북대학교 대학원 정보통신 공학과 (공학박사)

2007년 9월~현재 충북대학교 전기전자컴퓨터공학부 Post-doc

관심분야 : 데이터베이스 시스템, 데이터 마이닝, 생물정보학, 시스템 바이오

유 재 수

e-mail : yjs@chungbuk.ac.kr

1989년 전북대학교 컴퓨터공학과 (공학사)

1991년 한국과학기술연구원 전산학과 (공학석사)

1995년 한국과학기술연구원 전산학과 (공학박사)

1996년~현재 충북대학교 전기전자컴퓨터공학부 교수

관심분야 : 데이터베이스 시스템, 정보검색, 멀티미디어 데이터베이스, 분산객체 컴퓨팅, 생물정보학