생물학적 서열들에서 빈발한 연속 서열 패턴 마이닝

강태호○ 유재수
충북대학교 정보통신공학과
thkang@chungbuk.ac.kr, yjs@chungbuk.ac.kr

Mining Frequent Contiguous Sequence Patterns in Biological Sequences

Tae Ho Kang○ Jae Soo Yoo
Dept. of Computer and Communication Engineering, Chungbuk National University

요 약

생물학적 서열 데이터는 크게 DNA 염기 서열과 단백질 아미노산 서열이 있다. 이들 서열은 일반적으로 많은 수의 항목들을 가지고 있어 그 길이가 매우 길다. 생물학적 데이터 서열들에는 보통 빈번하게 발생하는 부분 연속 서열들이 존재하는데 이들 서열들을 찾아내는 것은 다양한 서열 분석에서 유용하게 사용될 수 있다. 이를 위해 초기에는 Apriori 알고리즘을 기반으로 하는 순차패턴 마이닝 알고리즘들을 활용하는 방법들이 많이 제시되었다. 그중 PrefixSpan 알고리즘은 Apriori기반의 가장 효율적인 순차패턴 마이닝 기법이다. 하지만 이 알고리즘은 길이-1인 빈발 패턴들로부터 서열 패턴을 확장해나가는 방식으로 길이가 긴 연속 서열을 포함하는 생물학적 데이터 서열들에 대한 검색방법으로는 적합하지 않다. 최근에는 기존의 PrefixSpan방식을 이용하면서도 반복적인 처리과정을 줄인 MacosVSpan이 제안되었다. 하지만 이 알고리즘 또한 원본 데이터베이스보다 크기가 큰 별도의 프로젝션 데이터베이스를 사용함으로서 많은 비용부담이 발생하고 특히 길이가 긴 서열에 대해서는 더욱 효율적이지 못하다. 이에 본 논문에서 많은 양의 생물학적 데이터 서열들로부터 빈번한 연속서열을 고정길이 확장 트리를 이용하여 효과적으로 찾아내는 방법을 제안한다. 그리고 다양한 환경에서 실험을 통해 제안하는 방식이 MacosVSpan알고리즘에 비해 검색 성능이 우수함을 증명한다.

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. 이중 생물학적 서열들에 대한 비교는 대량의 생물학적 데이터 분석을 위한 대표적인 생물정보학 기술이라 할 수 있다

생물학적 서열 데이터는 크게 DNA 염기 서열과 단백질 아미노산 서열이 있다. 각 서열들은 수백 또는 수천 개의 항목들을 가지고 있어 그 길이가 매우 길다 그리고 이들 서열들은 보통 유전적인 변형 또는 변이로부터 보존된 영역이나 특정 패턴들을 서열 안에 포함하고 있는데 이러한 서열 패턴들은 계통발생학적 근거로 활용될 수도 있으며 기능과 밀접한 관계를 가지는 모티프 등을 찾는데도 활용될 수 있다.

모티프 및 도메인에 대한 생물정보학적 검색 방법은 크게 두 가지로 나뉜다. 첫째 주어진 서열 데이터 집합으로부터 집합을 대표할 서열이나 추정되는 모티프 및 도메인을 찾아내는 것이다. 둘째는 주어진 서열이 특정 모티프나 도메인을 갖고 있는지 여부를 기존 공개용 데이터베이스를 대상으로 검색하는 것이다

이중에서 특정 서열 패턴을 찾는 것은 첫 번째에 해당하며, 두 개 이상의 서열들로부터 빈번하게 발생하는 최대 길이의 연속 패턴을 발견하고자 하는 것이다[1][2][3]. 두 개 이상의 DNA 염기 서열이 주어졌을 때 이들의 생물학적 상관관계를 파악하기 위해 서열 패턴들이 이용될 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 발견할 수 있는 알고리즘을 제안한다 그리고 제안하는 알고리즘의 우수성을 보이기 위해 다양한 환경에서 실험을 수행한다. 본 논문의 구성의 다음과 같다. 먼저 2장에서 최근까지의 관련연구를 설명하고 문제점을 제시한다 그리고 3장에서 문제점을 해결하기위해 새롭게 제안하는 알고리즘을 설명하고 제안하는 알고리즘의 특징을 분석

"이 논문은 2007년 정부(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임" (지방연구중심대학육성사업/충북BIT연구중심대학육성사업단)

한다. 4장에서는 기존에 제안된 알고리즘들과의 성능평가를 통해 본 알고리즘의 우수성을 설명하고 마지막으로 5장에서 결론을 맺는다.

2. 관련 연구

두 개 이상의 서열들로부터 빈번하게 발생하는 최대길이의 부분 연속 서열을 찾기 위해 많은 연구들이 수행되어져 왔다. 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 검색 시간이 기하급수적으로 증가한다는 문제가 있다. 최근에는 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 반복적으로인 데이터베이스에 접근 및 프로젝션 데이터베이스를 생성해야하기 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 데이터베이스를 사용하고 최소 지지도를2로 가정한다.

표 1. DNA 서열 데이터베이스

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

표 1의 DNA 서열 데이터베이스에는 길이7의 최대 연속 서열이 2번 이상 존재함을 알 수 있다. 이를 찾기 위해 MacosVSpan 알고리즘에서는 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 1은 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 1의 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 최대 길이 서열들에 대해서 접미사 트리를 생성함으로서 최종적인 최대길이 서열을 찾을 수 있다.

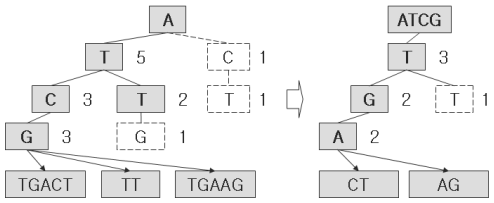

그림 1. 고정길이 span method

이 알고리즘은 PrefixSpan 방식을 사용하면서도 서열 패턴을 확장하기위해 반복적인 수행과정을 많이 감소시켰다. 하지만 MacosVSpan 알고리즘의 경우도 프로젝션 데이터베이스를 별도로 생성하고 이를 이용한다는 문제점이 있다. 프로젝션 데이터베이스는 원본 데이터베이스에 비해 상당히 큰 데이터양을 차지하는데 서열의 길이가 길어질수록 프로젝션 데이터베이스의 양은 기하급수적으로 늘어난다는 문제가 있다. 이에 대한 자세한 설명은 4.1절의 프로젝션 데이터베이스의 문제점에서 다룬다.

3. 제안하는 알고리즘

3.1 연속 서열 패턴 마이닝 알고리즘

알고리즘

```
입력 : 서열집합 S(S1, S2.. SN), 고정길이 W, 최소지지도 Min_Sup
출력 : 고정길이 접미어 확장트리 T, 최대길이 연속서열 후보 집합 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 2는 본 논문에서 제안하는 빈번한 최대길이 서열 패턴을 찾기 위한 알고리즘이다. 알고리즘은 크게 3가지 단계로 나뉜다. 먼저 첫 번째 단계에서는 주어진 서열들

로부터 고정길이 윈도우 길이의 부분 서열들을 읽어서 확장 트리를 구축한다. 두 번째 단계에서는 구축된 트리를 깊이우선 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 검색한다. 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.

알고리즘 설명을 위해 표 1과 같은 데이터베이스가 있다고 가정한다. 그리고 고정길이를 4로 최소 지지도를 2로 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 3은 서열 데이터를 고정길이 만큼 읽는 것을 보이고 있다.

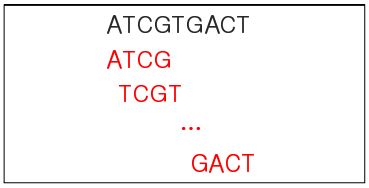

그림 3. 고정길이 스캔

전체 서열을 그림 3과 같은 방식으로 지정된 고정길이 만큼 읽으며 확장 트리를 구축하면 그림 4와 같다. 트리의 각 노드는 카운트를 포함하고 있어 부분 서열의 중복 횟수를 유지한다.

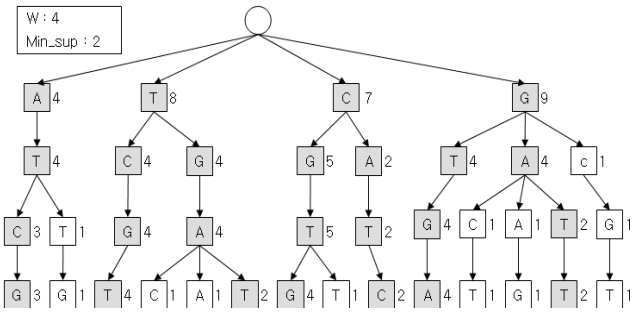

그림 4. 고정길이 확장 트리 구축

고정길이 확장 트리가 구축되면 곧바로 트리를 순회하면서 카운트 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 사실을 기반으로 한다.

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 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 2와 같다. 표 2는 길이가 4인 부분 연속 서열로부터 최대 확장 가능한 서열들을 예측한 것이다.

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	TCGTGAT
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			

표 3은 서열들에서 일정 지지도를 만족하는 서열들을 실제 검사한 결과이다. 표 2와 표 3을 통해 두 결과가 거의 유사함을 알 수 있다. 그리고 이를 통해 최대길이의 후보 서열 또한 유추가 가능함을 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 더 이상 확인하지 않는다.

3.2 알고리즘 분석

본 논문에서 제안하는 알고리즘은 다음과 같은 특징을 가진다. 첫 번째, 한 번의 데이터베이스 접근과 고정길이 확장트리를 구축으로 여러 값의 최소 지지도를 유연하게 수용할 수 있다. 두 번째, 트리를 구성하는 고정길이를 크게 할수록 상대적으로 검색 성능을 높일 수 있다. 세 번째, 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 서열

뿐만 아니라 항목 수가 많은 단백질 아미노산 서열 및 기타 다차원의 서열데이터에 대해서도 적용 가능하다.

4. 성능평가

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 4는 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 용량은5000(Byte), 8932(Byte)이다. 그리고 각 서열들로부터 핵산의 염기 A, C, G, T에 대한 각각의 프로젝션 데이터베이스를 생성한 결과 데이터 용량의 차이를 보이고 있다.

표 4. 프로젝션 데이터베이스의 용량 비교

데이터 / 데이터 타입	원본 데이터	A-프로젝션 데이터	C-프로젝션 데이터	G-프로젝션 데이터	T-프로젝션 데이터
무작위 서열	5000	66529	61286	64734	59901
DNA 서열	8932	1034951	920142	967499	1015148

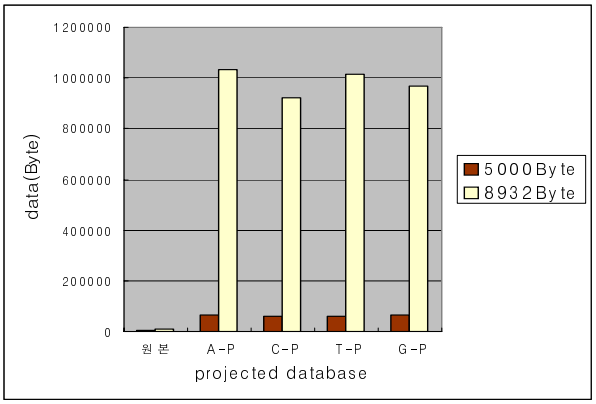

그림 5. 프로젝션 데이터베이스의 용량비교

그림 5는 표 4에서 보이는 차이를 보다 확인하기 쉽도록 그래프를 이용해 차이를 비교했다. 표 4와 그림 5는 원본 서열 데이터의 용량이 커질수록 프로젝션 데이터베이스의 용량이 기하급수적으로 증가함을 보인다. 또한 서열의 길이가 길어질수록 프로젝션 데이터베이스의 용량은 더욱 증가하게 된다. 프로젝션 데이터베이스는 검색 성능에 많은 영향을 미친다. 따라서 본 논문에서는 이러한 문제점을 해결하기 위해 프로젝션 데이터베이스를 생성하지 않고도 빈번하게 발생하는 부분 연속 서열 검색이 가능하도록 하였다.

4.2 검색 성능 및 메모리 사용량

그림 6은 기존의 MacosVspan 알고리즘에 비해 제안하는 알고리즘 검색 성능이 우수함을 보인다.

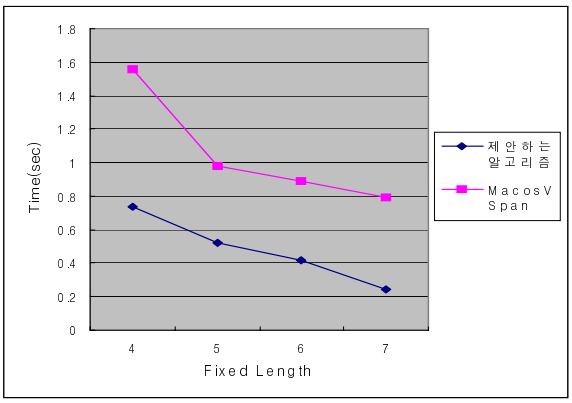

그림 6. 알고리즘 검색성능 비교

MacosVspan 알고리즘의 경우 길이가 긴 서열 데이터로부터 최대길이의 공통부분 서열을 검색하기 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 할수록 도출되는 후보 서열이 적어지기 때문에 이러한 부정확한 데이터를 획기적으로 줄일 수 있다. 이러한 사실은 그림 6의 고정길이의 변화에 따른 검색성능의 변화를 통해 확인할 수 있다.

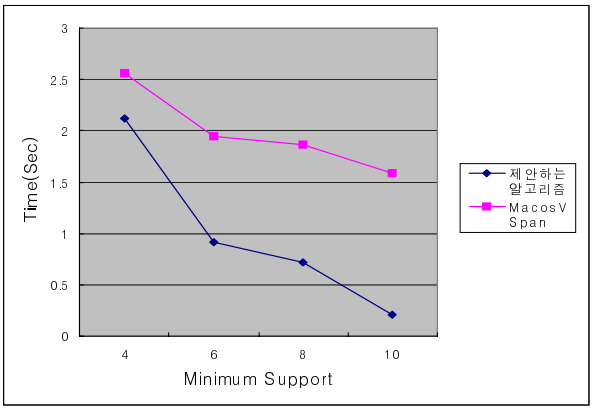

그림 7. 최소지지도 변화에 따른 검색 속도

그림 7은 빈발 지지도에 따른 검색 결과를 기존방식과 비교한 것이다. 앞에서 설명한 실제 랜덤 데이터를 이용해 고정길이 5인 트리를 구축하였다. 처음 최소 지지도가 4인 부분에서는 기존방식과 검색 속도 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 검색 시간을 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 않다. 마지막으로 지지도가 10인 부분에서는 지지도를 만족하는 길이 5의 서열이 극히 적어 후보 집합 확장 및 산출 없이 트리 구축 및 검색만으로 최종 결과를 도출하는 경우를 보이고 있다.

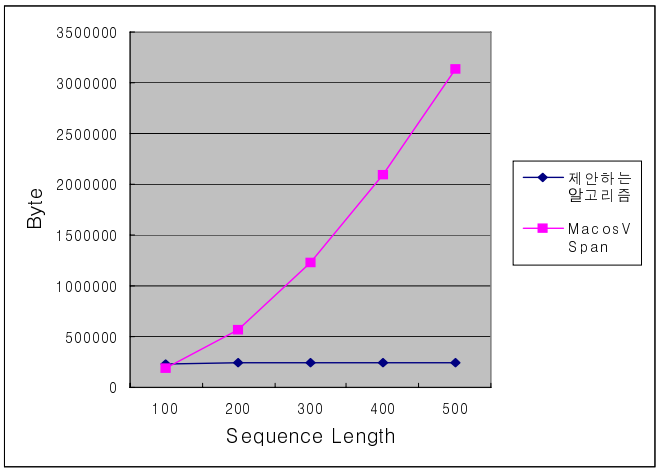

그림 8. 두 알고리즘에 사용된 메모리양

그림 8은 그림 6에서 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 구축하기 때문에 조금 더 많은 메모리 사용량을 요구한다. 하지만 이러한 현상은 비교하는 서열의 길이가 작을 경우에 나타난다. MacosVspan 알고리즘은 서열의 길이가 길어질수록 접미어의 크기기기하급수적으로 커지기 때문에 많은 메모리를 필요로 한다. 반면 제안하는 알고리즘의 경우 고정길이의 확장 트리는 최대 크기 이후 더 이상 증가하지 않는다.

<표 5> 고정길이 변화에 따른 메모리 사용량

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

마지막으로 표 5는 제안하는 알고리즘에서 고정길이 변화에 따른 메모리 사용량을 나타낸다. 고정길이를 5에서부터 10까지 변화시켜가며 실제 랜덤 데이터를 이용해 트리를 구축해보고 이때 필요한 메모리양을 측정했다.

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 크기다.

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 생성을 없앰으로서 서열데이터에 대한 비교 및 검색 성능을 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.

6. 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 Pattenrs: 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,