DNA 서열의 위치 정보를 이용한 효율적인 유사성 검색 알고리즘

정인선*, 박경욱, 임형석
전남대학교 전산학과
e-mail : isjung0@moiza.jnu.ac.kr

An Efficient Algorithm for Similarity Search using Positional Information of DNA Sequences

In-Seon Jeong*, Kyoung-Wook Park, Hyeong-Seok Lim
Dept. Computer Science, Chonnam National University

요 약

유전자 데이터베이스의 서열의 길이가 수백만에서 수백억 정도의 대용량 텍스트이기 때문에 기존의 Smith-Waterman 알고리즘으로 정확한 서열의 유사성을 검색하는 것은 매우 비효율적이다. 따라서 빠른 유사성 검색을 위해 데이터베이스에 저장된 문자열에 대해 특정 길이의 모든 부분문자열에 나타나는 문자의 출현 빈도를 이용한 휴리스틱 방법들이 제안되었다. 이러한 방법들은 질의 서열과 일치될 가능성이 높은 후보들만을 추출한 후 이들 각각에 대하여 질의 서열과의 일치 여부를 조사하므로 빠르게 유사성 검색을 할 수 있다. 그러나 이 방법은 문자의 출현 빈도만을 사용하므로 서로 다른 서열을 같은 서열로 취급하는 단점이 있어 정확도가 Smith-Waterman 알고리즘에 비해 떨어진다. 본 논문에서는 문자가 부분문자열에 나타나는 위치 정보를 포함하여 문자의 출현빈도를 인덱싱함으로써 질의 처리를 효율적으로 수행하는 알고리즘을 제안한다. 실험결과 제안된 알고리즘은 문자 빈도만을 사용하는 알고리즘에 비해 5~15%정도 정확성이 향상되었다.

1. 서 론

새롭게 발견된 서열의 생물학적 특성을 파악하기 위한 기본적인 방법은 그것과 서열의 배열이 유사한 다른 서열의 특성으로부터 유추하는 것이다[1]. 서열의 유사성을 이용하여 새롭게 발견된 서열의 역할, 진화 과정, 화학적 구조 등을 유추한다. 서열의 유사성을 찾는 문제는 이미 특성이 판명된 서열 데이터베이스로부터 주어진 질의 서열과 배치가 유사한 서열들을 검색하는 것이다. 서열에는 돌연변이의 발생과 오염의 발생이 가능하므로 서열 검색 연산으로써 완전 일치 질의(exact match query)보다는 근사 일치 질의(approximate match query)가 널리 사용된다[2].

지금까지 밝혀진 유전자의 서열은 수백만에서 수백억 염기쌍(base pairs, bp)으로 이루어진 대용량 텍스트이다. 때문에 전체 유전자에서 특정 유전자의 위치를 찾는 것과 같은 문자열 일치를 메모리 상주(in-momory)방식으로 처리하는 것은 비효율적이다.

서열의 검색 연산의 기초적인 방식은 주어진 두 서열들 간의 최적의 배열(optimal alignment)을 찾아주는 Smith-Waterman (S-W) 알고리즘[3]이나 BLAST[4,5]를 이용하는 것이다. 이 알고리즘들은 길이가 각각 n과 m인 두 서열들의 배열을 계산하는데 $O(nm)$의 시간을 요구하므로, 서열 비교 속도가 느리다. 또한 S-W 알고리즘은 검색 시 전체 데이터 서열을 디스크로부터 액세스해야 하는 단점이 있다. 따라서 일련의 문자열로 구성된 대용량 유전자 데이터베이스를 대상으로 메모리 크기에 제한을 받지 않으며, 작은 저장 공간을 사용하는 인덱스 기법을 기반으로 하여 효율적인 질의 처리 기법이 요구 되어진다.

데이터베이스에 저장된 서열들 중에서 질의 서열과 가장 유사한 서열들을 검색하는 질의로 1)질의 서열과 일정한 유사성 허용오차를 만족하는 모든 서열들을 검색하는 영역 질의(range query)와 2)질의 서열과 가장 유사한 k-개의 서열을 검색하는 k-최근접 질의(k-Nearest Neighbor)가 있다[6]. 영역 질의는 적절할 결과 값을 얻기 위해 유사성 거리와 거리의 오차에 대한 정확한 정의를 요구한다. 유사성 허용오차를 작은 값으로 설정하면 적은 검색 결과가 산출 되지만, 허용오차를 큰 값으로 설정하면 많은 검색 검색결과가 산출된다. 두 경우 모두 필요로 하는 정보를 얻을 수 없거나 유용하지 못한 정보를 얻게 된다. 따라서 본 논문에서는 유사성 검색에 많이 사용되는 k-최근접 질의를 고려한다.

효율적인 서열의 유사성 검색을 위해 데이터베이스에 저장된 문자열에 대해 특정 길이의 모든 부분문자열에 나타나는 문자의 출현 빈도를 이용한 인덱싱 방법들이 제안되었다. 이러한 방법들은 질의 서열과 일치될 가능성이 높은 후보들만을 추출한 후 이들 각각에 대하여 질의 서열과의 일치 여부를 조사하므로 빠르게 유사성 검색을 할 수 있다. 그러나 이 방법은 문자의 출현 빈도만을 사용하므로 서로 다른 서열을 같은 서열로 취급하는 단점이 있어 정확도가 Smith-Waterman 알고리즘에 비해 떨어진다. 제안된 기법에서는 서열 내에서 각 문자의 출현 빈도와 더불어 위치 정보를 다차원의 벡터로 추출하여 이를 R-트리[7]와 같은 다차원 공간 인덱스에 저장한다.

2. 관련연구

기존에 알려진 서열을 대상으로 유사성 검색을 통해 연관성 있는 서열의 전부나 일부분을 알아내는 것은 생물학자들이 가

장 빈번히 사용하는 방식이다.

가장 광범위하게 사용되는 알고리즘은 Smith-Waterman[3]과 BLAST[4,5]이다. 순차적 검색을 기반으로 하는 대표적인 유사성 검색 기법으로 S-W[3]를 들 수 있다. S-W 알고리즘은 동적 프로그래밍을 이용하여 전체 서열에서 서열 검색을 수행하는데 연산의 정확성을 보장하지만, 많은 계산량을 필요로 하기 때문에 실제적인 사용에 있어 제한적이다.

BLAST[4,5]는 S-W 알고리즘이 전체 서열들을 질의 서열과 비교하는 단점을 해결하기 위해 여과 및 정제(filtering and refinement) 기법을 채택했다. 즉, 여과 단계에서 질의 서열과 일치될 가능성이 높은 후보들만을 빠르게 검색한 후, 정제 단계에서 이들 각각에 대하여 실제로 질의 서열과 일치되는지의 여부를 확인하는 것이다. 이 알고리즘은 휴리스틱 기반 알고리즘에 의하여 고속의 서열 검색을 제공하지만, 관련 데이터베이스가 주기억 장치에 적재되어야 하며, 검색 속도가 데이터베이스 사이즈에 비례하고, 워드의 고정 길이에 따라 검색 결과의 정확도가 영향을 받는 점 등이 문제점으로 지적된다.

[8,9]에서는 k-디퍼런스 질의(k-difference query)를 처리하기 위해 MR-Index를 제안하였다. k-디퍼런스 질의는 대치, 삽입, 삭제 등을 k회 이하로 적용함으로써 질의 서열과 일치되는 데이터 서열들의 집합을 찾는 연산이다. 이 기법에서는 먼저 2^n 형태로 표현되는 다양한 길이의 슬라이딩 윈도우들을 데이터 서열로 추출한 후, 각 슬라이딩 윈도우에 대해 서열들의 출현 빈도를 계산하여 다차원의 벡터 형태로 변환한다. 그 다음 동일한 길이의 슬라이딩 윈도우로부터 작성된 벡터들을 모두 모아서 R-트리[7]에 저장한다. 이 검색 기법은 작은 저장 공간을 사용하는 인덱스 기법을 기반으로 하여 대용량 유전자 데이터베이스를 대상으로 메모리 크기에 제한을 받지 않지만, 윈도우 내의 문자 출현 빈도만을 이용해 서열을 벡터로 변환하므로 동일한 벡터를 가지는 서로 다른 서열이 같은 서열로 취급될 수 있어 오류율이 높은 단점을 지닌다.

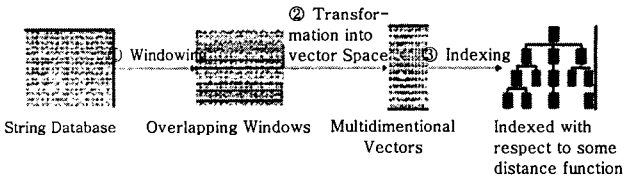
3. 제안한 알고리즘

본 장에서는 [9]에서 제안한 인덱스 기반의 웨이블렛(wavelet) 변환 방식을 이용한 검색 기법의 각 문자의 빈도수가 같은 서로 다른 서열을 동일한 서열로 취급하는 문제를 해결하는 효율적인 알고리즘을 기술한다.

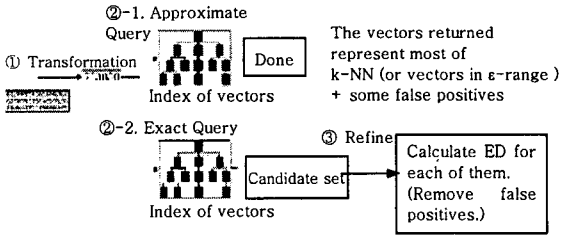
3.1 데이터 전처리와 인덱싱

데이터 전처리와 인덱싱을 처리하는 과정은 [그림 1]과 같다. 인덱스를 구성하기 위해 먼저 유전자 데이터베이스 T에서 일정 크기를 갖는 슬라이딩 윈도우를 위치시켜 윈도우를 생성한 후, 각 윈도우에 대하여 윈도우 내에 출현하는 각 문자의 위치 정보를 n차원 벡터로 추출하여 R-트리와 같은 다차원 공간 인덱스에 저장한다.

정의 1. 알파벳 $\sum=\{a_1, a_2, \ldots, a_\sigma\}$로 구성된 길이가 l인 문자열 S에 대해 N-grams의 출현 위치에 대한 벡터 $WPT\#N$을 다음과 같이 정의한다.



[그림 1] 데이터의 전처리와 인덱싱



[그림 2] 질의 실행

$$WPT\#N=[n_1, n_2, \ldots, n_{\sigma^N}]$$

여기서 n_i는 문자열 S내에 출현하는 문자 a_i의 출현 위치에 대한 가중치들의 합이다.

문자열 S에서 i번째 문자의 가중치(W_i)는

$$W_i = l + (i \bmod (l/2))$$

으로 l은 문자열 S의 길이이다.

예를 들어 유전자 데이터 $\sum$={A, C, G, T}에 대해 문자열 S={ACTCTAGC}일 경우 $WPT\#1$={17, 31, 10, 18}이고, $WPT\#2$={0, 8, 9, 0, 11, 0, 0, 20, 0, 10, 0, 0, 8, 10, 0, 0}이다.

벡터 크기는 문자열 길이에 상관없이 4^N이다. N=2일 경우 두 개의 문자가 한 쌍을 이루기 때문에 4^2=16이 된다. 즉, N=1일 때 $\sum$={A, C, G, T}, N=2라면 $\sum$={AA, AC, AG, AT, CA, CC, ..., TT}이다.

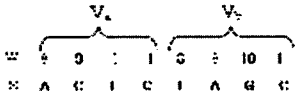
N의 크기가 증가할수록 변환된 벡터의 크기는 기하급수적으로 증가한다. 만약 N=3일 경우 $WPT\#3$의 벡터 크기는 4^3=64가 되어 많은 저장공간을 필요로 한다.

정의 2. 알파벳 $\sum=\{a_1, a_2, \ldots, a_\sigma\}$로 구성된 길이가 l인 문자열 S에 대해 N-grams의 출현 위치에 대한 벡터 $WWT\#N$은 다음과 같이 정의한다.

$$WWT\#N=[V_a, V_b]$$

여기에서 $V_a = WPT\#N(S[0..l/2-1])$와 $V_b = WPT\#N(S[l/2..l-1])$이다.

예를 들어 유전자 데이터 $\sum$={A, C, G, T}에 대해 문자열 S={ACTCTAGC}일 경우 $WWT\#1$={8, 20, 0, 10, 9, 11, 10, 8}이 된다.



[그림 3] WWT#1의 벡터 변환

각 문자의 출현 빈도만을 이용해 벡터 형태로 변환할 경우 동일한 벡터값을 가지는 서로 다른 서열들이 발생할 수 있으므로 각 문자의 위치에 가중치를 부여함으로써 이러한 발생을 감소시킬 수 있다.

R-트리를 이용하여 벡터 *WWT#N*으로 표현된 각 윈도우의 벡터들은 다차원 공간 인덱스로 저장된다.

3.2 질의 실행

질의 실행을 처리하는 과정은 [그림 2]와 같다.

3.1에서 제안한 기법을 이용하여 질의 서열 역시 σ^N차원의 벡터로 변환한다.

문자열 S와 질의 서열의 σ^N차원의 벡터를 각각 u와 v라 하자. 두 벡터 사이의 거리를 계산하기 위한 알고리즘(*WWD#N*)은 다음과 같다.

알고리즘 1. *WWD#N(u, v)*

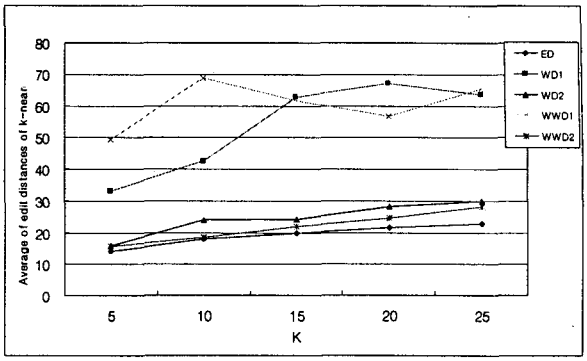
```
WWD#N(N-gram의 WWT#N u, v)
positiveDistance := 0
negativeDistance := 0
for all dimensions u_i, v_i
        if  u_i>v_i    positiveDistance += u_i-v_i
        else           negativeDistance += v_i-u_i
return
        max(positiceDistance, negativeDistance)/N
```

이 알고리즘의 시간복잡도는 $O(\sigma^N)$이다.

R-트리를 이용하여 다차원 공간 인덱스로 저장된 벡터들은 알고리즘 1에 의해 질의 서열과의 거리를 계산하여 유사성이 높은 벡터를 k의 개수만큼 추출한다. 근사 일치 질의(approximate match query)는 추출된 벡터들을 최종 결과값으로 한다. 이 방법에서는 유사성이 높지 않은 서열들이 포함 되어질 수도 있다. 반면 완전 일치 질의(exact match query)는 추출된 벡터들을 후보자로 선정하여 이들 각각에 대해서만 실제로 질의 서열과 일치되는지의 여부를 확인한다.

4. 성능평가

본 논문에서 제안한 알고리즘의 효율성을 테스트 해보기 위해 [8]에서 제안된 웨이블렛 변환에 의해 각 문자의 출현 빈도를 벡터로 변환한 방식(*WD#N*)과 비교하였다. 제안된 알고리즘은 JAVA로 구현하여 펜티엄4 2.8GHz에서 18번 인간 염색체(약 $7.6*10^7$bp)에 대해 테스트하였으며, 질의 서열은 같은 샘플에서 임의로 선택하였다. 기존의 실험과 같이 윈도우의 길이 w=200, 윈도우 이동의 크기 Δ=10로 설정하였다. 테스트 결과 [그림 4]과 같이 *WD#2*에 비해 *WWD#2*이 5~15%정도 정확하게 서열을 검색하였다. 또한 동적 프로그래밍 기법을 기반으로 하여 두 서열 사이의 최적의 거리(optimal distance) S-W 알고리즘에 비해 10~20%정도의 오차를 보였다.



[그림 4] 시뮬레이션 결과

5. 결론

본 논문에서는 효율적인 유사성 검색을 위해 인덱스 구조를 기반으로 하여 질의 처리 기법을 제안하였다. 제안된 알고리즘은 문자열 내의 각 문자의 출현 빈도와 더불어 위치정보를 σ^N차원의 벡터 형태로 변환하여 다차원 공간 인덱스에 저장한다. 이는 각 문자의 출현 빈도만을 이용해 벡터 형태로 변환할 경우 동일한 벡터값을 가지는 서로 다른 서열들이 같은 서열로 취급될 수 있으므로 각 문자의 위치에 가중치를 부여하여 좀 더 서열들을 효율적으로 검색할 수 있는 기법이다. 또한 완전 일치 질의(exact match query)에서 서열과 일치될 가능성이 높은 후보들만을 빠르게 검색할 경우 사용될 수 있다.

참 고 문 헌

[1] C. Gibas and P. Jambeck, Developing Bioinformatics Computer Skills, O'reilly and Associates Inc., 2001.

[2] D. W. Mount, Bioinformatics: Sequence and Geonome Analysis, Cold Spring Harbor Laboratory Press, 2001.

[3] T. Smith and M. Waterman, "Identification of common molecular subsequences," Journal of Molecular Biology, Vol. 147, pp. 195-197, 1981.

[4] S. Altschul, W, Gish, W. Miller, E. Myers, and D. Lipman, "Basic Local Alignment Search Tool," Jouranl of Molecular Biology, Vol. 215, pp. 403-410, 1990

[5] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: A new generation of protein database search programs," Nucleic Acids Research, 25(17), 1997

[6] D. A. White and R. Jain, "Similarity Indexing with the SS-tree," Conf On Data Engineering, pp. 516-523, 1996

[7] Guttman, "R-Trees, A dynamic index strcture for spatial searching," ACM SIGMOD, pp. 47-57, 1984

[8] T. Kahveci and A.K Singh, "An efficient Index Structure for String Databases," VLDB, pp. 351-360, 2001

[9] O. Ozturk and H. Ferhatosmanoglu, "Effective Indexing and Filtering for Similarity Search in Large Biosequence Database," IEEE, 2003