The Korean Journal of Microbiology, Vol. 35, No. 2, June 1999, p. 115-120

진화 알고리즘을 사용한 복수 염기서열 정렬

김 진[1*] · 송민동[2] · 최흥식[3] · 장연아[3]

건국대학교 자연과학대학 전산과학과[1]*, 분자생물학과[2],
한림대학교 컴퓨터공학부[3]

3개 이상의 DNA 혹은 단백질의 염기서열을 정렬하는 복수 염기서열 정렬(multiple sequence alignment)은 염기서열들 사이의 진화관계, gene regulation, 단백질의 구조와 기능에 관한 연구에 필수적인 도구이다. 복수 염기서열 정렬을 얻기 위한 기존의 방법은 progressive pairwise alignment와 같이 빠른 실행시간 내에 만족할 만한 복수 염기서열 정렬을 제공하는 방법과, 최적의 복수 염기서열 정렬을 제공하나 실행시간이 상대적으로 긴 dynamic programming과 같은 방법 등이 있다. 본 논문에서는 진화 알고리즘을 사용하여 기존의 방법에서 제공하는 복수 염기서열 정렬을 짧은 시간 내에 보다 개선된 복수 염기서열 정렬을 획득하게 하는 방법을 제시하였으며, 진화 알고리즘의 구성내용을 설명하였으며, 실제의 염기서열을 사용하여 이 방법의 장점을 보였다.

KEY WORDS □ **multiple sequence alignment, genetic algorithm, dynamic programming, sequence comparison**

생물학 역사상 가장 중요한 프로젝트의 하나인 Human Genome Project의 기본적인 목표는 인체의 게놈과 생명체의 유전자의 염기서열의 인식을 목표로 하고 있다. 이 프로젝트에 의해 발생되는 엄청난 양의 염기서열 관련 데이터는 의약과 생물학 분야에 절대적인 영향력을 미치고 있으며 이러한 추세는 더욱 심화될 것이라 예상된다. 이러한 염기서열 관련 데이터를 처리하여 중요한 생물학적 정보를 얻기 위해서는 전산학의 도움이 필수적이다. 전산학에서 염기서열은 스트링으로 간주된다. 본 논문에서는 게놈 프로젝트에서 파생된 가장 중요한 문제 중에 하나인 복수 염기서열 정렬(multiple sequence alignment) 문제에 대하여 논한다(3-5, 7, 20).

염기서열 정렬은 단백질(protein), DNA 및 RNA의 생물학적 분석에 관련된 문제이다. 생물학자들은 두 개 혹은 그 이상의 유전자 염기서열들을 어떤 종류의 자연적인 생물학적 측정 단위(natural biological metric)를 사용하여 최소값을 가지는 염기서열 정렬을 획득하려 한다. 최소값을 가지는 염기서열 정렬은 미지의 염기서열(unknown sequence)의 확인을 위한 유전자 데이터베이스 검색, 유사한 단백질 분자구조와 관련된 패턴인식, 염기서열의 기능 및 기능과 진화에 관한 중요한 정보를 획득하기 위하여 사용된다.

염기서열 정렬기법 중, 두 개의 염기서열에 관한 정렬기법은 아주 잘 연구되어 있으며, dynamic programming기법을 사용하여 효율적으로 최적해를 가진 정렬을 얻을 수 있다(10. 16, 18, 20). 세 개 이상의 복수 개의 염기서열을 정렬하는 주요한 방법으로는 progressive pairwise alignment 방법과(6, 15), dynamic programming을 사용하여 최적해(optimal) 혹은 최적해에 가까운(near optimal) 값을 가지는 복수 염기서열 정렬을 얻는 방법이 있다(13). Progressive pairwise alignment에 의한 방법은 짧은 실행시간을 사용하여 만족할 만한 복수 염기서열 정렬을 생산한다. 그러나 이 방법은 획득되는 복수 염기서열 정렬이 최적 비용을 가짐을 보장해주지 못한다 반면에 dynamic programming을 사용하여 복수염기서열 정렬을 얻는 방법은 최적, 혹은 최적에 가까운 복수 염기서열 정렬을 제공하나, 정렬하려 하는 염기서열의 개수가 증가함에 따라 필요한 실행시간도 지수 함수적으로 증가하기 때문에 4~7개 이상의 염기 서열정렬에는 효율적이지 못하며, 특정한 측정 단위에는 사용될 수 없는 등의 단점이 있다. 복수 염기서열 정렬에 사용되는 또 다른 방법은 simulated annealing(11-13), 진화 알고리즘(genetic algorithm) 등과 같은 추정 통계적(stochastic)인 최적화 방법을 사용하는 것이다(9, 17). Simulated annealing과, 진화 알고리즘 등이 3개 이상의 염기서열 정렬에 사용되어 왔는데, dynamic programming의 단점인 오랜 실행시간 문제를 해결할 수 있으나, 이 방법들 또한 많은 계산 시간이 요구되며 최적의 값을 가지는 복수 염기서열 정렬의 획득을 보장하지 못한다.

본 논문에서는 기존의 방법을 사용하여 획득된 복수염기 서열의 질을 진화 알고리즘을 사용하여 개선하는 방법을 제시하도록 한다 이를 위하여 기존의 알고리즘에서 획득된 복수 염기서열 정렬을 이용하는 진화 알고리즘을 제시하였으며, 실험 데이터로 실제 단백질 염기서열을 사용하였다. 실험 결과, 짧은 계산시간으로 기존의 알고리즘에서 획득된 복수 염기서열 정렬을 개선하여 최적에 가까운 비용을 가지는 복수 염기서열 정렬을 찾을 수 있었다.

본 논문에서는 복수 염기서열 정렬에 대하여 보다 자세히 설명하며, 유전자 알고리즘에 대하여 정의하며, 복수 염기서열 문제에 어떻게 진화알고리즘을 적용하는가에 대해 설명한다. 다음으로 실제 단백질 염기서열 데이터를 사용하여 실험한 결과를 분석하며 마지막으로 결론을 맺는다.

*To whom correspondence should be address
Tel : 0441-840-3619, Fax : 0441-851-4169
E-mail : jinkim@kcucc.kku.edu

복수 염기서열 정렬 (multiple sequence alignment) 문제

DNA, RNA 및 단백질들의 염기서열들은 유전에 관한 중요한 정보를 가지고 있다. 염기서열은 DNA의 경우 Σ={a, t, g, c}, RNA인 경우 Σ={a, u, g, c}, 단백질의 경우 Σ={W, Y, F, V, L, I, M, K, R, H, Q, E, D, N, G, A, P, T, S, C}의 알파벳으로 이루어진 유한한 길이의 스트링이라 정의할 수 있다. 예를 들면 스트링 "**gcctaccgagcc**"는 DNA의 염기서열이라 할 수 있으며, 염기서열 순서확인이란 유전 물질로부터 염기서열의 글자를 읽어내는 작업이라 정의할 수 있다.

복수 염기서열 정렬 문제는 계산 생물학(computational biology) 내에서도 가장 중요한 문제중의 하나이다. 복수 염기서열 정렬은 염기서열들 간의 element들의 대응관계를 알아내는 것과 관련이 있다. 복수 염기서열 정렬은 생물학적 데이터 분석에 있어 중요한 작업으로 미지의 염기서열의 확인을 위한 데이터베이스 검색, 유사한 분자구조와 관련된 패턴인식, 염기서열의 기능 및 기능과 진화에 관한 중요한 정보를 획득하기 위하여 사용된다. 이러한 염기서열의 정렬을 위해 정렬의 양호성(goodness)을 측정할 수 있는 비용함수(f)가 필요하다. 특정 염기서열들의 집합에 대해, 최적 정렬(optimal alignment)은 최소 비용(minimum cost)을 가지는 정렬이다. 최소 비용을 가진 정렬을 얻기 위해, 각 염기서열들의 원소들의 사이에 매치(match), 삽입(insertion), 교체(substitution), 삭제(deletion) 등이 사용된다.

Fig. 1은 세 개의 염기서열의 정렬의 한 예이다. Fig. 1(a)는 정렬되지 않은 세 개의 염기서열들을 보여준다. Fig. 1(b)는 동일한 세 개의 염기서열을 기본 알파벳과 null('-')을 사용하여 세 개의 염기서열들 간의 유사성을 보다 잘 나타낼 수 있도록 표현되었다.

Fig. 1(b)의 1, 2, 5, 7, 8열은 동일한 알파벳으로 매치(|)되었으며, 첫 번째 염기서열의 3번째 요소는 교체됨을 보여주며, 세 번째 염기서열의 6번째 열은 삭제되었음을 보여주며, 첫 번째 열의 4번째 열은 삽입되었음을 나타낸다. 삽입과 삭제를 표현하기 위해 기본 알파벳이외에 '-'가 사용되었다. null의 생물학적인 의미는 하나의 염기서열에서 삽입이 발생했거나, 혹은 다른 염기서열에서 삭제가 발생했다는 것이다 복수 염기서열 정렬문제는 다음과 같이 정의할 수 있다.

정의 1

·알파벳(alphabet) Σ는 문자들과 null('-')로 이루어진 유한 집합이다.

```
          1 2 3 4 5 6 7 8                    1 2 3 4 5 6 7 8
염기서열 1 t g c t g c t c    염기서열1    t | g c t g c t c
염기서열 2 t g a g c t c      염기서열2    t | g a - g c t c
염기서열 3 t g a g t c        염기서열3    t | g a - g - t c
```

Fig. 1. Examples of multiple sequence alignment (a) roultiple sequence alignment which dose not represent the simularity among the sequences (b) A multiple sequence alignmet which shows the similarity among the sequences. Good multiple sequence alignments represent the similarities among the aligned sequences. Matches (|) represent same alphabets in the same rows.

·염기서열(sequence)은 문자로 이루어진 유한한 길이의 스트링이다.

·염기서열 S_1= s11s12 s1n1, S_2= s21s2 s2n2,. , Sn= sk1sk...sknk은 각각 길이 n_1, n_2,..., n_k의 n개의 입력 염기서열들이며 염기서열 정렬은 null이 패드된(padded) 동일한 길이 *l*의 의사염기서열 S_1'= s11's12'..s1l',S_2'= s21's22' s2l', .., S_n'= sn1'sn2' .snl'으로 이루어진 집합이다. 이때 패드된 의사염기서열(pseudo-sequence) S_i'로부터 null들을 제거하면 원래의 염기서열 Si를 얻을 수 있다. 이러한 염기서열 정렬을 다음과 같은 이차원 배열로 나타낼 수 있다

S_1'= s11's12' . s1l'
S_2'= s21's22'. s2l'
...
S_n'= sn1'sn2'...snl'

이때 null들의 위치와 길이 n과 관련되어 서로 다른 많은 종류의 복수 염기서열 정렬이 얻어질 수 있다.

복수 염기서열 정렬 문제(multiple sequence alignment)는 정렬에 대한 최적도의 기준이 주어졌을 때, 최적 비용(optimal cost)을 가지는 복수 염기서열 정렬을 찾는 문제이다.

진화 알고리즘

복수 염기서열 정렬문제는 NP-complete 문제군에 속하는 문제로 알려져 있다(8). 진화 알고리즘은 자연계의 진화과정을 흉내낸 계산모델로서 NP-complete 문제 군에 속하는 문제들에 대한 최적 혹은 최적에 가까운 해를 제공하는 알고리즘이다. 진화 알고리즘은 개체(individual)라 불리는 후보 해(candidate solution)들의 모집단(population)들을 운영한다[5]. 전형적으로, 초기 모집단은 무작위 개체들로 구성된다. 이후, 개체들은 그들의 상대 적합도(fitness)에 기초하여 모집단에서 제거되거나 재생산된다. 새로운 개체들은 기존의 개체들의 모집단에 다양한 연산자(operator)들을 적용하여 생성된다. 개체의 연속적인 모집단을 세대(generation)이라 한다. 진화 알고리즘에서 사용되는 연산자들은 다음과 같다.

연산자

선택(selection) 각 개체들의 적합도가 측정된다. 개체들은 그들의 적합도에 따라 재생산 된다. 이때 재생산에 적용하는 방식은 문제마다 같지 않다.

교배(crossover) 모집단으로부터 두 개의 개체가 선택되며 개체들 내의 상응하는 위치로부터 substring들이 교환된다. 새로이 생성된 개체들이 다음 세대의 모집단에 추가된다.

변이(mutation) 변이는 한 개체의 기본 요소를 변화시켜 새로운 개체를 만들어내는 연산자이다. 변이율에 의해 개체내의 각 요소가 변화되는 확률이 조정된다. 새로이 만들어진 개체가 다음 세대의 모집단에 추가된다.

이러한 연산자들을 일정, 혹은 가변적인 확률에 의해 적용하여 조금 더 우수한 개체, 혹은 최적해를 찾는다. 진화의 과정은 최초의 개체군에서 시작하여 새로운 개체군들을 반복적으로 생

성하며 세대교체를 반복하는 일정의 집단에 의한 탐색과정이다 세대교체시 적합도가 높은 개체들이 다음 세대의 개체군 생성에 더 많은 영향력을 미쳐 우성 인자들을 유전시키도록 한다.

진화 알고리즘의 단계

진화 알고리즘의 단계는 다음과 같다.

1) 무작위 초기 모집단에서 시작한다.

2) 모집단 내의 각 후보해법에 대해 최적해로서의 적합도를 측정한다.

3) 후보해법들에 연산자를 적용하여 새로운 후보해법을 만들어낸다.

4) 새로운 후보해법을 평가한 후 모집단에 추가한다.

5) 종료조건을 만족할 때까지 3-4단계를 반복하고, 만족하면 최적 적합도를 가진 해법을 출력한다.

복수 염기서열 정렬을 위한 진화 알고리즘

비용 함수

염기서열 정렬에 사용되기 위한 비용 함수(cost function)는 정렬의 질(quality)에 대한 명백한 측정수단이어야 한다. 즉 작은 비용을 가지는 복수염기서열은 큰 비용을 가지는 복수염기서열보다 생물학적 현상을 잘 표현하여야 한다. Altschul(1, 2)은 복수 염기서열 정렬을 위한 몇 가지의 비용 함수를 분류하였는데, 이들 비용함수들은 교체 비용(substitution cost)과 갭 비용(gap cost)으로 구성된다.

교체 비용 교체 비용은 DNA나 단백질 분자를 대표하는 글자들을 정렬하는데 드는 비용이다. 본 논문에서 사용된 교체 비용은 SP(sum of pairs) 교체비용(1)이다. SP 교체비용은 n개의 염기서열이 정렬되었을 때, n(n-1)/2개의 쌍의 교체비용의 합이다. 즉 복수 염기서열 정렬 A의 교체비용은

$$\text{교체비용(A)} = \sum_{i=1}^{n-1} \sum_{j=i}^{n} \text{교체비용}(S_i', S_j')$$

으로 계산될 수 있다.

갭 비용 갭 비용은 연속된 null('-')의 갯수에 부과되는 비용이다. Alstchul은 생물학적으로 자연스러운 갭 비용함수로써 natural gap costs를 제안하였다(1, 2). 본 논문에서는 natural gap costs를 사용한다.

어떤 복수 염기서열 정렬 A에 대한 비용 함수 f는 f(A)=교체비용(A)+갭 비용(A)로 정의할 수 있으며, 복수 염기서열 정렬 문제는 최소의 비용을 가지는 염기서열 정렬을 찾는 것을 목표로 하는 문제이다.

초기 모집단의 생성

진화 알고리즘의 처음 단계는 초기 모집단의 생성이다 먼저 progressive pairwise alignment 방법과 같은 빠른 경험적인(heuristic) 알고리즘을 사용하여 최초의 해를 얻는다. 이때 얻어진 최초의 복수 염기서열은 일반적으로 최적의 비용을 가지는 복수 염기서열 정렬이 아니다. 이 해에 들어있는 null('-')들의 위치를 변경하여 50개체로 구성된 초기 모집단을 생성한다. 초기해에 들어있는 null의 초기위치가 기록되며 null의 위치는 초기위치 L(임계값)내에서만 이동이 가능하다. 이는 빠른 경험적인

```
염기서열1 AKMKQIGGA--MSG      AKMKQIGGA--MSG      AKMKQIGGA--MSG
염기서열2 AKMKQIGGA--MSG      AKMKQIGGA--MSG      AKMKQIGGA--MSG
염기서열3 AKMKK---IGGATG      AKMKK--I-GGATG      AKMKKI---GGATG
          (a)                 (b)                 (c)
```

Fig. 2. An example of swap operators. (a) Current alignment (b) New alignment generated by applying swap operator. left null(-) and 1 in sequence 3 are swapped.

알고리즘이 만들어내는 복수염기서열의 모양은 최적의 비용을 가지는 염기서열은 유사할 것이라는 가정에 의한 것이다. 이때 만일 dynamic programming에 의해 얻어진 복수 염기서열보다 개선된 복수 염기서열 정렬을 얻기 위해서는, dynamic programming에 의해 얻어진 해를 초기 해로 사용할 수도 있다. 이 해에 진화 알고리즘을 적용하였을 때에 해가 개선되지 않았을 때 이 해는 최적해라 할 수 있다. 이러한 방법에 대한 이론적인 근거는 Kim의 논문에 잘 해석되어있다(11).

개체의 평가 및 새로운 세대의 생성

새 세대를 만들기 위하여 각 개체의 적합도를 평가한다. 이 적합도는 비용 함수에 따라 각 정렬에 대한 비용을 계산하여 얻는다. 비용이 작은 정렬일수록 적합도는 높아진다. 한 세대의 개체들 가운데 적합도가 낮은 50%만이 후손으로 교체된다.

새로운 후손들은 부모를 선택한 후 그들을 변경하여 생성한다. 부모를 선택할 때 부모의 적합도와 관련된 확률에 의한다. 부모를 변경하기 위하여 몇 가지의 연산자가 사용되었다.

변이 연산자 Kim은 simulated annealing을 복수 염기서열 정렬에 적용하였다(11). 이때 사용된 연산자는 swap연산자였다. 본 논문에서는 이 연산자와 유사한 swap연산자를 변이 연산자로 사용하였다(Fig. 2).

swap1 연산자는 기본적으로 정렬에 들어 있는 null들을 이웃한 알파벳과 바꾸어 null의 위치를 달리해 주는 연산자이다. 선택된 정렬내의 염기서열과 null을 무작위로(random) 선택하고, 이 null을 이웃한 알파벳과 교체하여 준다. Fig. 2(a)의 기존 개체에서 염기서열3과 염기서열3 내의 연속된 null의 오른쪽 알파벳인 I가 랜덤하게 선택되어 왼쪽의 null과 교체되어 (b)와 같은 새로운 개체가 생성되었다. swap2 연산자는 Fig. 2(a)의 기존 개체의 연속된 null을 이동시켜 Fig. 2(c)와 같은 새로운 개체를 만들어낸다. swap1 연산자와 swap2 연산자의 상대적인 사용빈도는 90%이상으로 swap2연산자가 크다.

교배 연산자 교배 연산자는 모집단으로부터 두 개의 다른 정렬을 선택연산자에 의해 두 개의 부모 정렬을 선택하여 부모 정렬내의 염기서열들을 무작위로 선택하여 혼합해 하나의 새로운 정렬을 만들어낸다. Fig 3(a)는 선택된 두 개의 부모이고, 두 부모를 혼합하였을 경우 (b)와 같은 새로운 염기서열 정렬이 만

```
염기서열1 AKMKQIGGA--MSG      AKMKQI--GGAMSG      AKMKQIGGA--MSG
염기서열2 AKMKQIGGATG---      AKMKQIGG----ATG     AKMKQIGG----ATG
염기서열3 AKMKQIGGATG---      AKMKQIGG---ATG      AKMKQIGGATG----
          (a)                 (b)                 (c)
```

Fig. 3. An example of crossover operator. (a) and (b) are parent alignments. (c) New alignment generated by crossover operator. sequence1, sequence 3 in alignment (a) and sequences 2 in alignment (b) are combined into new alignment (c).

```
begin
        /* 입력 염기서열, 비용함수 구성 */
        initialize();
        /* 빠른 heuristic알고리즘에 의한 최초해 획득, 초기 모집단 생성*/
        pop* initializer(int pot_size)
        /* 각 개체의 Fitness Value를 계산 */
        evaluation(pop* population)
        while(generation<max_generation){
                /* 주어진 모집단에 대해 확률에 의한 교차 연산자를 수생하여
                  새로운 세대로 세대를 교체함 */
                pop* crossover(pop* population)
                evaluation(pop* population)
                /* 주어진 모집단의 각 개체에 돌연변이율에 따라 mutation연산자를 수행한다 */
                mutation(pop* population)
        }
end
```

Fig. 4. Genetic algorithm in this study.

들어진다.

선택연산자 이전 세대에서 다음 세대의 자손을 만들기 위한 부모를 선택하는 것은 우수한 적합도를 가진 개체에 높은 확률을 부여하여 두 개의 부모 개체를 선택한다. 본 논문에서 실험한 염기서열 정렬의 적합도 값은 비용함수에 의해서 계산된다.

진화 알고리즘은 추정 통계적인 방법이므로 이론적으로 무한한 계산시간을 적용하여도 이 방법이 최적 비용을 가진 복수 염기서열 정렬을 생산할 수 있다는 보장이 없다. 이것은 다른 추정 통계적인 기법들에도 적용된다. 본 연구에서는 10000세대이후 프로그램을 종료하도록 하고 그 과정에서 얻어진 최소의 비용을 가진 복수 염기서열 정렬을 출력하도록 하였다.

실험을 위해 사용된 프로그램의 개략적인 알고리즘

본 절에서는 위에서 설명한 초기해 구성 및 연산자를 이용하여 구현한 프로그램의 전반적인 알고리즘을 순서도로 요약한다 (Fig. 4). 본 실험은 UNIX상에서 ANSI C가 사용되었다.

실험 및 결과

본 논문에서 사용된 실험 데이타들은 chymotrypsin, trypsin과 elastase family에 속하는 염기서열들이었다. 정렬한 염기서열의 개수는 3~10개이었으며, 길이는 200~300이었다.

```
IIGGVESIPHSRPYMAHLDIVTEKGLRVICGGFLISRQFVLTAAHC
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHC
IVNGEEAVPGSWPWQVSLQDKTGF---HFCGGSLINENWVVTAAHC
IVGGYTCGANTVPYQVSL--NSGY---HFCGGSLINSQWVVSAAHC
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHC
VVCCTRAAQGEFPFMVRL--SMG------CGGALYAQDIVLTAAHC
```

(a)

```
IIGGVESIPHSRPYMAHLDIVTEXGLRVICGGFLISRQFVLTAAHC
IVGGTNSSWGEWPWQVSLQVKLTAQ-RHLCGGSLIGRQWVLTAAHC
IVNGEEAVPGSWPWQVSLQDKTG----FHFCGGSLINEWWVVTAAHC
IVGGYTCGANTVPYQVSLNSG------YHFCGGSLINSQWVVSAAHC
VVGGTEAQRNSWPsQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHC
VVGGTRAAQGEFPFMVRLSMG---------CGGALYAQDIVLTAAHC
```

(b)

Fig. 5. (a) Multiple sequence alignment generated by progressive pairwise alignment, (b) Multiple sequence alignment generated by genetic algorithm.

```
GDSGGPLLCAGV------AHGIVSYG
GDSGGPLVCKHN-GMWRLVGITSWG
GDSGGPLVCKKN-GAWILVGIVSWG
GDSGGPVVCSGK------LQGIVSWG
GDSGGPLHCLVN-GQYAVHGVTSFV
GDSGGPMFRKDNADEWIQVGIVSWG
```

(a)

```
GDSGGPLLCAG-----VAHGIVSYG
GDSGGPLVCKHN-GMWRLVGITSWG
GDSGGPLVCKKN-GAWILVGIVSWG
GDSGGPVVCSG-----KLQGIVSWG
GDSGGPLHCLVN-GQYAVHGVTSFV
GDSGGPMFRKDNADEWIQVGIVSWG
```

(b)

Fig. 6. (a) Multiple sequence alignment generated by dynamic programming (b) Multiple sequence alignment generated by genetic algorithm.

실험 1

Fig. 5(a)는 빠른 heuristic algorithm에 의해 획득된 염기서열 정렬이다. 이 염기서열 정렬을 진화 알고리즘의 초기 해로 사용하였다. 세대 수는 100으로 하였으며, 돌연변이율은 각 세대에서 40%로 하였다.

Fig. 5(a)의 비용은 9737이었다. Fig. 5(a)에 진화 알고리즘을 적용하여 Fig 5(b)와 같은 개선된 복수 염기서열 정렬을 얻을 수 있었다. 이때 Fig. 5(b)의 비용은 9726이었다.

실험 2

실험 2는 dynamic programming에 의해 얻은 결과를 진화 알고리즘의 입력 염기서열로 사용하였다. Fig. 6(a)는 dynamic programming에 의해 획득된 염기서열의 일부며 (b)는 염기서열 (a)를 진화 알고리즘에 입력으로 적용하여 얻어진 결과이다.

염기서열 (a)의 비용은 5038이며, (b)의 염기서열은 5030으로 진화 알고리즘을 적용한 결과 보다 낮은 비용의 염기서열 정렬을 얻을 수 있었다. Dynamic programming은 최적값을 가진 염기서열 정렬을 제공하지만 특별한 경우에 있어 최적비용을 가진 염기서열 정렬을 제공하지 않는다. 위의 경우가 그 특별한 경우인데, 한 염기서열내의 연속된 null에 다른 염기서열의 null이 완전히 포함되어 있는 경우에 최적의 염기서열정렬을 제공

```
IVGGTNSSWGEWPWQVSLQVKLTAQR-HLCGGSLIGHQWVLTAAHCFDGLPLQDVWRIYS
IVNGEEAVPGSWPWQVSLQDKTFG---HFCGGSLINENWVVTAAHCGVT----TSDVVVA
IVGGYTCGANTVPYQVSLN--SGY---HFCGGSLINSQWVVSAAHCYKS------GIQVRL
VVGGTEAQRNSWPSQISLQYRSGSSWAHTCGGTLIRQNWVMTAAHCVDR--ELTFRVVVG
VVGGTRAAQGEFPFMVRLS--MG------CGGALYAQDIVLTAAHCVSG----SGNNTSI

GILNLSDITKDTPFSQIKEIIIHQNYKVSEGNH--DIALIKLQAPLNYTEFQKPICLPSK
GEFDQGSSSEKIQKLKIAKVFKNSKYNSLTINN--DITLLKISTAASFSQTVSAVCLPSA
GEDNINVVEGNEQFISASKSIVHPSYNSNTLNN--DIMLIKLKSAASLNSRVASISLPTS
GEHNLNQNNGTEQYVGVQKIVVHPYWNTDDVAAGYDIALLRLAQSVTLNSYVQLGVLPRA
TATGGVVDLQSAVKVRSTKVLQAPGYNGTG--K--DWALIKLAQPINQPTLKIATTTAYN

GDTSTIYTNCWVTGWGFSK-EKGEIQNILQKVNIPLVTNEECQ-KRYQDYKITQRMVCAG
SDDFAAGTTCVTTGWGLTRYTNANTPDRLQQASLPLLSNTNCK--KYWGTKIKDAMICAG
CASAG--TQCLISGWGNTKSSGTSYPDVLKCLKAPILSDSSCK-SAYPG-QITSNMFCAG
GTILANNSPCYITGWGLTR-TNGQLAQTLQQAYLPTVDYAICSSSSYWGSTVKNSMVCAG
QGTFT------VAGWGANR-EGGSQQRYLLKANVPFVSDAACF-SAYGNELVANEEICAG

Y-KEGGKDACKGDSGGPLVCKHNG-MWRLVGITSWGE--GCARREQPGVYTKVAEYMDWI
---ASGVSSCMGDSGGPLVCKKNG-AWTLVGIVSWGS--STCSTSTPGVYARVTALVNWV
Y-LEGGKDSCQGDSGGPVVCSGK------LQGIVSWGS--GCAQKNKPGVYTKVCNYVSWI
---GNGVRSCQGDSGGPLHCLVNG-QYAVHGVTSFVSRLGGCNVTRKPTVFTRVSAYISW
YPDTGGVDTCQGDSGGPMFRKDNADEWIQVGIVSWGY--GCARPGYPGVYTEVSTFASAI

ELKTQSS
QQTLAAN
KQTIASN
INNVIAS
ASAARTL
```

Fig. 7. Multiple sequence alignment generated by genetic algorithm.

하지 못한다(1). 그러나 진화 알고리즘을 사용할 경우 이러한 문제가 발생하지 않기 때문에 최적의 비용을 가진 염기서열 정렬을 제공할 수 있다.

실험 3

실험 3은 5개의 완전한 염기서열을 dynamic programming을 사용하여 정렬을 얻은 후 이 정렬을 진화 알고리즘의 입력 해로 사용하였다. Fig. 7은 진화 알고리즘에 의해 획득된 염기서열 정렬을 나타낸다.

이때 dynamic programming에 의해 획득된 염기서열의 비용은 36014이었으며, 진화 알고리즘에 의해 획득된 최종 염기서열의 비용은 35845이었다. Dynamic Programming에 의해 획득된 염기서열정렬에서 null들이 존재하는 부분들만을 선택하여 진화 알고리즘을 적용하는 방법으로 진화 알고리즘의 수행시간을 줄일 수 있었다. 실험 1, 실험 2 의 경우는 수 초의 실행시간으로 위의 정렬을 획득할 수 있었으며, 실험 3의 경우 10분 정도가 소요되었다.

결 론

본 논문에서는 진화 알고리즘을 사용하여 최적비용을 가진 복수염기서열 정렬 문제를 다루었다. 실험 1, 2, 3에서와 같이 진화 알고리즘을 사용한 복수 염기서열 정렬은 만족할만한 결과를 제시하여 주고 있다. 진화 알고리즘의 장점은 복수 염기서열 정렬에 관한 어떠한 비용 함수라도 사용할 수 있다는 점이다. 부모개체를 선택한 후 연산자들을 적용하여 완벽한 새로운 개체를 만들어내기 때문에 비용 함수를 적용하는데 필요한 모든 정보를 얻을 수 있다. Dynamic programming의 경우에 있어서는 알고리즘의 한계성 때문에 특정 비용함수를 복수 염기서열 정렬에 사용할 수 없다. 진화 알고리즘의 또 다른 장점은 기존의 알고리즘에 의해 획득된 염기서열 정렬을 개선할 수 있다는 것이다.

본 연구에서는 두 개의 가장 기본적인 연산자들만이 사용되었다. 그러나 이 두 개의 기본 연산자들 이외에도 보다 효율적인 연산자가 있을 수 있다. 본 연구에서는 경험적인(heuristic) 알고리즘에 의해 만들어진 해를 초기 해의 원천으로 사용하기 때문에 local minimum에서 빠져 나오지 못할 수 있다. 이를 위한 보다 효율적인 연산자를 연구 중에 있다. 현재 이 알고리즘을 Visual C++을 사용하여 PC상에서 구현하여 생물학자들에게 보다 쉽게 사용할 수 있도록 구현 중에 있다.

감사의 말

본 연구는 한국학술진흥재단의 '97 공모과제(학제간 연구) 지원으로 수행된 연구결과임.

참고문헌

1. **Altschul, S.F.** 1989. Gap costs for multiple sequence alignment *Journal of Theor. Biol.* **138**, 297-309.
2. **Altschul, S.F. and D.J. Lipman.** 1989. Trees, stars, and multiple biological sequence alignment. *SIAM J. Appl. Math.* **49**, 153-161.
3. **Bacon, D.G. and W.F. Anderson.** 1986. Multiple sequence alignment. *J. Mol. Biol.* **191**, 153-161
4. **Chan, S.C., A.K.C. Wong, and D.K.Y. Chiu.** 1992. A survey of multiple sequence comparison methods. *Bull. Math. Biol.* **54**, 563-598.
5. **Carrillo, H. and D. Lipman.** 1988. The multiple sequence alignment problem in biology. *SIAM J. Appl. Math.* **48**, 1073-1082.
6. **Feng, D. and R.F. Doolittle.** 1987. Progressive sequence alignment as a prerequisite to correct phylogenetic trees *J. Mol. Evol.* **25**, 351-360.
7. **Fickett, J.W.** 1987. Fast optimal alignment. *Nucl. Acids Res.* **12**, 175-180.
8. **Garey, M.R. and D.S. Johnson.** 1979. Computers and Intractability. A guide to the theory of NP-completeness, W. H. Freeman. San Francisco, CA.
9. **Goldberg, D.E.** 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, New York.
10. **Gotoh, O.** 1982 An improved algorithm for matching biological sequences. *J. Mol. Biol.* **162**, 705-708.
11. **Kim, J., S. Pramanik, and M.J. Chung.** 1994. Multiple sequence alignment using simulated annealing. *Comp. Appl. Biosci.* **10**, 419-426.
12. **Kirkpatrick, S., C.D. Gelatt, and M.P. Vecchi.** 1983. Optimization by simulated annealing *Science.* **220**, 671-680.
13. **Lipman, D.J., S.F. Altschul, and J.D. Kececioglu.** 1989. Tool for multiple sequence alignment. *Proc. Natl. Acad. Sci.* USA. **86**, 4412-4415.
14. **Metropolis, M., M. Rosenbluth, A. Rosenbluth., A. Teller, and E. Teller.** 1953. Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087-3077.
15. **Miller, W.** 1993. Building multiple alignment from pairwise alignments. *Comput. Applicat. Biosci.* **9.** 11-17.
16. **Needleman, S.B. and C.D. Wunch.** 1970. A general method applicable to the search for similarities in the amino acid sequences of two proteins, *J. Mol. Biol.* **48**. 443-453.
17. **Notredame, C. and D.G. Higgins.** 1996. SAGA: sequence alignment by genetic algorithm *Nucl. Acids Res* **24**(8) 1515-1524.
18. **Sankoff, D.** 1972. Matching sequence under deletion-insertion constraints. *Proc. Natl. Acad. Sci.* USA. **64**. 4-6.
19. **Smith, T.F. and M.S. Waterman,** 1981 Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195-197.
20 **Waterman, M.S.** 1984. General methods of sequence comparison. *Bull. Math. Biol.* **46**, 473-500.

(Received January 5, 1999/Accepted March 22, 1999)

ABSTRACT : Multiple Sequence Aligmnent Genetic Algorithm

Jin Kim[1]*, Min-Dong Song[2], Hongsik Choi[3] and Yeonah Chang[3] (Department of computer science[1], Department of molecular biology, Konkuk Univ.[2], Div. of computer engineering Hallym University[3])

Multiple Sequence Alignment of DNA and protein sequences is a important tool in the study of molecular evolution, gene regulation, and protein structure-function relationships. Progressive pairwise alignment method generates multiple sequence alignment fast but not necessarily with optimal costs. Dynamic programming generates multiple sequence alignment with optimal costs in most cases but long execution time. In this paper, we suggest genetic algorithm to improve the multiple sequence alignment generated from the current methods, describe the design of the genetic algorithm, and compare the multiple sequence alignments from our method and current methods.