퍼지 및 지능시스템학회 논문지 2004, Vol. 14, No. 1, pp. 105-111

Traveling Salesman Problem을 해결하기 위한 DNA 코딩 방법을 적용한 DNA 컴퓨팅

DNA Computing Adopting DNA coding Method to solve Traveling Salesman Problem

김은경* · 윤효근* · 이상용**

Eun-Gyeong Kim, Hyo-Gun Yun, and Sang-Yong Lee

* 공주대학교 컴퓨터 공학과

** 공주대학교 정보통신공학부

요 약

Traveling Salesman Problem(TSP)을 해결하기 위해 DNA 컴퓨팅이 사용되고 있다. 그러나 현재의 DNA컴퓨팅을 TSP에 적용하였을 때, 정점과 정점사이의 가중치를 효율적으로 표현할 수 없다.

본 논문에서는 TSP의 정점과 정점 사이의 가중치를 효율적으로 표현하기 위해 DNA 컴퓨팅 기법에 DNA 코딩방법을 적용한 ACO(Algorithm for Code Optimization)를 제안한다. 우리는 ACO를 TSP에 적용하였고, 그 결과 ACO는 Adleman의 DNA 컴퓨팅 알고리즘보다 가변길이의 DNA 코드와 간선의 가중치를 효율적으로 표현할 수 있었다. 또한 ACO는 Adleman의 DNA 컴퓨팅 알고리즘 보다 탐색 시간과 생물학적 오류율을 50%정도 줄일 수 있었으며, 빠른 시간 내에 최단 경로를 탐색할 수 있었다.

Abstract

DNA computing has been using to solve TSP (Traveling Salesman Problems). However, when the typical DNA computing is applied to TSP, it can't efficiently express vertices and weights of between vertices.

In this paper, we proposed ACO (Algorithm for Code Optimization) that applies DNA coding method to DNA computing to efficiently express vertices and weights of between vertices for TSP. We applied ACO to TSP and as a result ACO could express DNA codes which have variable lengths and weights of between vertices more efficiently than Adleman's DNA computing algorithm could. In addition, compared to Adleman's DNA computing algorithm, ACO could reduce search time and biological error rate by 50% and could search for a shortest path in a short time.

Key Words : Traveling Salesman Problem, DNA 컴퓨팅, DNA 코딩 방법, DNA 염기 서열

1. 서 론

DNA 컴퓨팅은 1994년 Adleman 이후에 조합 최적화 문제들을 이용하여 DNA 염기 서열의 디자인 및 합성 과정에 대하여 결과를 평가하고 있다. DNA는 10^{21}개의 DNA 염기를 가지며, 10억 terabits의 정보저장 능력을 지니고 있다. 또한 DNA 수용액속에는 6×10^{23}개의 분자를 가지고 있어 연산시 초병렬적인 정보처리가 가능하다[1, 2].

1998년에 조합 최적화 문제 중 TSP를 이용하여 DNA 서열을 디자인한 Narayanan과 Zorbalas은 Adleman의 DNA 컴퓨팅 알고리즘에서 DNA 서열로 표현된 간선에 일정한 가중치를 적용하여 DNA 코드를 생성하는 규칙을 제안하였다[3]. 그러나 이 생성 규칙을 이용하여 설계된 DNA 서열은 실제 DNA가 가지고 있는 가중치 값을 제대로 반영하지 못한 상태에서 DNA 서열 최적화를 이루었다. 특히 가중치에 비례해서 간선의 길이가 결정되기 때문에 마지막 가중치를 표현하기 위해서는 앞의 간선보다 매우 긴 DNA 서열이 필요하게 되어 매우 비효율적이다. 또한 가중치를 실수로 표현하기 어렵다는 단점도 가지고 있다.

그 후 Gonnet과 Korotensky는 TSP를 해결하기 위해 Dynamic Programming을 이용하여 간선에 대한 가중치 문제를 해결한 모델을 제시하였다[4]. 이 모델은 DNA 이중 구조에 대한 거리의 합을 이용하여 DNA 염기 서열들을 진화형 트리로 표현하였다. 그리고 각 말단 노드간에 거리값을 Dynamic Programming으로 계산하였다. 하지만 정점의 수가 증가하게 되는 경우 진화형 트리의 단점은 탐색 공간이 커진다는 것이다. 또한 각 정점간의 간선이 실제 가중치를 정확하게 반영하지 못한다는 것이다.

따라서 본 논문에서는 DNA 컴퓨팅의 서열 디자인에서 발생되는 문제점을 해결하기 위하여 DNA 컴퓨팅 알고리즘에 DNA 코딩 방법을 적용한 ACO(Algorithm for Code Optimization)를 제안한다. ACO는 가변길이의 DNA 서열과 간선의 가중치를 효율적으로 표현할 수 있기 때문에, 적절한 DNA 서열을 생성하여 최단 거리의 경로를 빠른 시간 내에 발견할 수 있다.

접수일자 : 2003년 7월 15일
완료일자 : 2003년 12월 30일
감사의 글 : 이 논문은 두뇌한국21사업에 의하여 지원되었음.

2. 관련 연구

2.1 Traveling Salesman Problem

TSP는 n개의 도시에 대하여 각 도시 사이의 거리가 주어질 때, 출발 도시에서 시작하여 모든 도시를 단 한번만 방문하고, 출발 도시로 되돌아오는 최단 길이의 경로를 찾는 문제이다. 다시 말하면 각각의 도시들에 대한 이동 경로가 순열로 주어질 때, i번째 도시에서 다음 도시와의 유클리드 거리의 합이 최소가 되는 경로를 선택하는 것이다. 따라서 TSP의 탐색 공간은 가능한 모든 경로의 집합 ($T_1, T_2, \ldots, T_{n!}$)으로, 이 중에서 거리가 가장 짧은 것이 해가 된다[8].

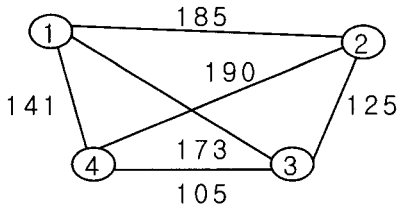


그림 1. TSP 그래프
Fig. 1. Graph of TSP

[그림 1]은 간단한 4개 도시의 TSP를 나타낸 것으로 최단 경로는 1 -> 4 -> 3 -> 2 -> 1 이며, 거리 값은 556이다. 또한 탐색 공간의 크기는 4!로 나타낸다.

2.2 DNA 컴퓨팅

DNA 컴퓨팅은 실제 생체 분자인 DNA나 RNA와 같은 살아 있는 세포를 응용한 기술이다. DNA는 4개의 염기인 A(Adenine), T(Thymine), C(Cytosine), G(Guanine)가 2중 나선 구조로 구성되어 있다. 이들 염기에 대용량 데이터를 저장할 수 있는 메모리 기능을 가지고 있으며, 정해진 규칙에 위해 상호 보완적인 방식의 Watson-Crick 결합을 하고 있다[1]. 그리고 복잡한 염기 조합의 패턴은 하나의 유전 정보를 담고 있으며, 인체내에서 자연 발생하는 효소에 의해 읽혀지고 있다. 효소는 생물학 실험 방법들과 함께 DNA 컴퓨팅의 연산자로 사용되고 있다.

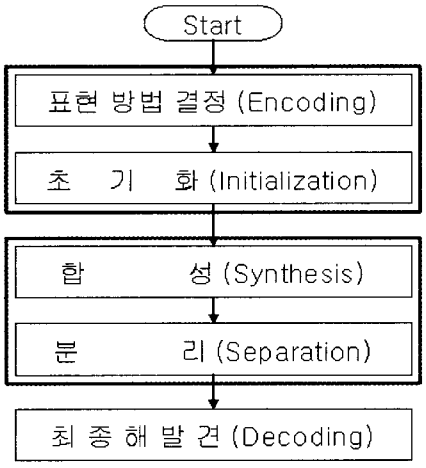


그림 2. Adleman의 DNA 컴퓨팅 알고리즘
Fig. 2. Adleman's DNA Computing algorithm

[그림 2]는 Adleman의 DNA 컴퓨팅 알고리즘을 나타낸 것이다. Adleman은 주어진 문제에 맞게 DNA 코드로 표현하고, 생성된 DNA 코드를 단 한번의 합성과 분리 과정을 거쳐 조합, 최적화 문제를 해결하였다[2].

이러한 DNA 컴퓨팅의 특징을 살펴보면 매우 낮은 에너지로 작동되기 때문에 많은 에너지가 필요없다. 그리고 나노 수준의 막대한 병렬성을 이용하여 NP-complete에 효과적인 접근이 가능하게 되었으며, 계산 속도와 정보의 저장 및 처리 효율에서도 우수함을 보이고 있다[2][3].

2.2 DNA 코딩 방법

DNA 코딩 방법은 1995년에 Yoshikawa가 처음으로 제시한 것으로, 변형된 형태의 유전자 알고리즘이다. 이는 DNA를 이용한 선택, 재생, 교배, 돌연변이 연산자를 사용한다[6][7].

DNA가 하나의 아미노산으로 해석되기 위해서는 3개의 염기서열이 필요하며, 이것을 생물학적인 용어로 코돈(codon)이라고 한다. 이것을 아미노산 번역표에 따라 총 64가지의 아미노산으로 해석된다. 하지만 중복된 것을 제외한 20개의 아미노산만으로 해석된다.

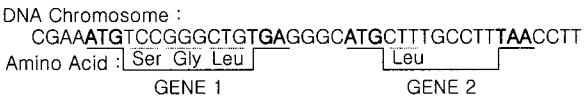


그림 3. DNA 염색체의 번역 예
Fig. 3. Example of DNA chromosome translation

[그림 3]에서 보는 것과 같이 염기서열은 start 코돈인 ATG에서 시작하여 stop 코돈인 TGA (TAA, TAG)에서 아미노산 번역이 끝나며, 코돈을 아미노산으로 해석함으로써 짧은 DNA 코드로에서도 많은 정보를 얻을 수 있다.

그림 4. 유전자 중복의 예
Fig. 4. Example of gene overlap

DNA 코딩 방법의 특징을 살펴보면 첫 번째로, [그림 4]에서 보는 것처럼 염색체의 중복을 효율적으로 표현할 수 있다는 것다. 두 번째로, 하나의 아미노산을 만드는 코돈이 여러 개이므로 지식 표현이 쉽다. 세번째로, 교차점이 임의로 주어지기 때문에 염색체의 길이가 가변적이라는 것이다.

이러한 특징들로 인해 긴 길이의 염기 서열이 아닌 적은 수의 아미노산 서열을 사용할 수 있고, 0과 1을 사용하는 유전자 알고리즘에 비하여 DNA 코딩 방법은 4가지 염기를 사용하여 코딩하기 때문에 해의 표현이 다양하다.

3. ACO

본 논문은 Adleman의 DNA 컴퓨팅 알고리즘을 개선하고 TSP를 이용하여 DNA 염기 서열의 최적화에서 발생되는 문제점을 해결하기 위한 ACO 알고리즘을 제안한다. ACO는 DNA 코딩 방법를 이용하여 TSP의 각 정점과 가중치를 포

합한 간선을 DNA 염기 서열로 표현하였으며, 특히 가중치를 포함한 간선은 실제 값을 효율적으로 반영하도록 한다. 그리고 DNA 코딩 방법의 연산자를 이용하여 DNA 합성과 분리 과정을 반복적으로 처리하도록 하여, 비효율적인 탐색으로 인한 시간과 노력이 낭비되는 것을 줄 일 수 있다. 또한 반복적인 합성과 분리 과정은 생물학적 실험에서 발생할 수 있는 오류율을 최소화함으로써 우수한 DNA 염기 서열을 탐색할 수 있다.

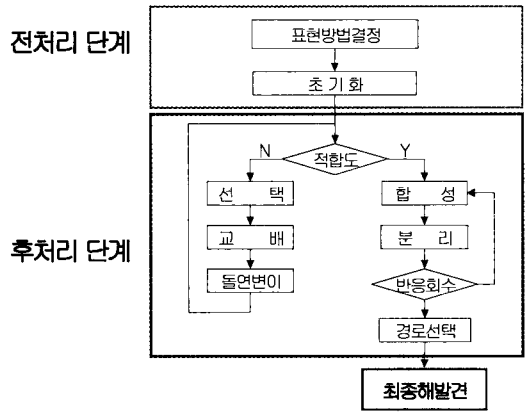


그림 5. ACO의 흐름도
Fig. 5. Flowchart of ACO

```
Initialize( N = Nodes,
            DNA_seq = (DNA sequence),
            S_codon = Start codon("ATG")
            codon_weight() )

Begin
    If (i Mod 2) = 1 Then
      i = 0
      Call Node_Make(N, DNA_seq, i)
    Else
      i = 1
      Call Node_Make(N, DNA_seq, i)
    End If
End

 Sub Node_Make(Node_t_No, DNA, Temp_Start_No)
    For L = 1 to Length(DNA) Step 3
    codon = Middle(DNA, i, 3)
        If codon = S_codon Then i = i + 1
        If (i*2-1) > Node_t_No Then Exit for
        If (i Mod 2) = 1 Then
          V(i) = V(i) + codon
        Else If( i Mod 2) = 0 Then
          W(i)=W(i)+codon + codon_weight()
        End If
    Next
 End Sub
```

그림 6. 정점과 가중치 표현 프로시저
Fig. 6. Procedure to express nodes and weights

[그림 5]에서 보는 것과 같이 ACO는 전처리 단계와 후처리 단계로 이루어져 DNA 염기 서열을 디자인한다. 두개의 단계를 거친 DNA 염기 서열 중 디자인이 우수한 것들만을 최종적으로 선택하게 된다. 먼저 ACO의 전처리 단계를 살펴보면, 표현 방법 결정 과정과 초기화 과정으로 나뉜다. 표현 방법 결정 과정에서는 주어진 염기 서열을 DNA 분자의 특성을 잘 반영할 수 있도록 DNA 코딩 방법을 이용하여 가변 길이의 정점과 가중치를 포함한 간선을 생성한다. 정점과 간선은 바로 표현할 수 없으며 이들을 DNA 염기 서열로 변환하기 위해서는 [그림 6]과 같은 프로시저를 이용한다.

먼저 정점과 가중치를 생성하기 위하여 모든 DNA 코드에 대해 start codon(ATG)의 위치를 파악하고, i번째 위치와 (i+2)번째 위치 전까지의 사이에서 (i+1)번째 위치 전까지를 정점으로 그 후를 가중치로 표현해 준다. 단, DNA code의 처음 위치가 start codon으로 시작되지 않을 경우 DNA 코드의 처음 부분부터 i번째 위치 전까지를 정점으로 한다.

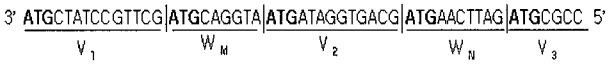


그림 7. 정점과 가중치의 예
Fig. 7. Example of nodes and weights

[그림 7]은 [그림 6]에 따라 정점과 가중치를 생성한 예이다.

```
Initialize( N = Nodes,
            DNA_seq = (DNA sequence),
            S_codon = Start codon("AT*, Not ATG")
            E_codon = End codon("TAA, TGA, TAG"))

Begin
   For i = 1 to N
    Sn = InString(V(i), S_codon)
    E(i) = Middle(V(i), Sn)
    En = InString(V(i+1), E_codon)
    If E = 0 Then
     E(i) = E(i) + W(i) +
          Middle(V(i+1), 1, Integer(Length(V(i+1))/2))
    Else
     E(i) = E(i) + W(i) + Middle(V(i+1), 1, En)
    End If
   E(i) = E(i)
   Next
End
```

그림 8. 간선 표현 프로시저
Fig. 8. Procedure to express edges

이렇게 표현된 정점을 이어주는 간선은 모든 DNA 코드에 대하여 [그림 8]의 프로시저를 따라 생성된다. 먼저 점정 Vi에서 처음으로 나타나는 AT*(ATT, ATC, ATA)를 $E_{(i)}$로 지정하고, $V_{(i+1)}$에서 처음으로 나타나는 stop codon인 TAA, TGA, TAG를 $E_{(i+1)}$로 지정하여 두 정점 사이를 간선으로 표현한다. 단, stop codon이 없을 경우에는 $V_{(i+1)}$의 염기 서열 1/2bp(base pair)를 간선으로 한다. 그리고나서 간선으로 설정된 DNA 코드를 가중치 W를 포함하여 가중치에 대해서만 상보 결합하여 표현한다.

그림 9. 간선의 예
Fig. 9. Example of an edge

[그림 9]는 [그림 8]에 따라 정점 V_1과 V_2를 이어주는 간선에 대한 표현 예이다.

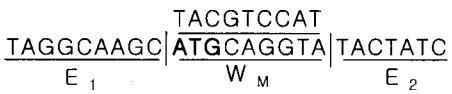


그림 10. 가중치를 포함한 간선의 예
Fig. 10. Example of an edge with a weight

가중치를 포함한 간선을 생성하기 위해서는 [그림 9]에 간선에 가중치 서열을 포함한 후 가중치에 대해서만 상보결합을 해주면 [그림 10]과 같다.

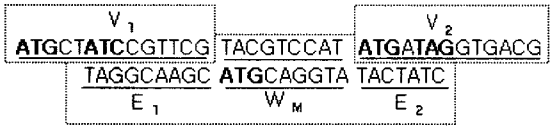


그림 11. 경로의 예
Fig. 11. Example of a path

[그림 11]은 정점과 가중치를 포함한 간선을 통합한 2중 가닥의 DNA 서열로 표현한 경로 생성의 예이다.

위와 같은 방법으로 정점과 가중치를 포함한 간선을 생성한 후 DNA 코드를 아미노산 번역표에 따라 아미노산 코드로 번역한다. 이처럼 표현 방법 결정 과정이 끝나면 다음 과정인 초기화 과정에서 결정한 표현 방법을 반영한 DNA 코드들을 생성한다.

ACO의 후처리단계를 살펴보면 적합도를 만족하지 않을 경우, DNA 코딩 방법의 연산자를 이용하여 적합도를 재평가한다. 만족하는 경우 반응횟수만큼 합성과 분리 과정을 거쳐 우수한 경로를 선택하여 최종해로 한다.

$$F_i = \left\{ \left| \sqrt{\frac{Ne_i}{Sv} - \frac{We_i}{Sw}} \right| \quad if \sqrt{\frac{Ne_i}{Sv} - \frac{We_i}{Sw}} \geq \Theta \right\} \quad otherwise\ is\ 0 \quad \text{식 (1)}$$

식(1)은 간선의 가중치 값을 구하는 식으로 간선 i의 수소결합 변환 함수값(Ne_i)과 간선 i의 실제 가중치(We_i), 전체 그래프에서 가중치의 합계(S_w), 전체 간선의 수소결합 총합(S_v), 실험을 통해 결정한 임계값(Θ)을 가지고 계산한다. 즉 A/T쌍과 G/C쌍의 수소결합 수를 각각 낮은 가중치와 높은 가중치를 가지는 간선에 포함시켜 가중치를 포함한 간선을 생성한다. 가중치 변환식을 이용하면 DNA 서열의 길이까지 조절하여 가중치를 표현할 수 있으며 가중치의 표현 범위가 훨씬 확장되어 짧은 서열을 가지고도 넓은 범위의 가중치를 표현할 수 있는 장점이 있다.

표 1. 아미노산 코드
Table 1. Amino acid code

Phe	16	Pro	3	His	15	Glu	13
Leu	7	Thr	5	Gln	11	Cys	6
Ile	8	Ala	1	Asn	9	Trp	19
Met	14	Tyr	18	Lys	12	Arg	17
Ser	2	Val	4	Asp	10	Gly	0

$$f(x) = \overline{x + k \mid \sin(32x) \mid}, 0 \leq x < \pi, k: \text{아미노산상수} \quad \text{식(2)}$$

본 논문에서 정점과 가중치를 포함한 간선의 적합도 평가는 [표 1]의 아미노산 코드를 적용하여 식(2)의 비례 선택법(roulette wheel)을 역함수로 평가한다. 그리고 잘못된 결합이나 결합 위치 이동과 같은 생물학 실험에서 발생할 수 있는 오류의 조건을 미리 제거한다. 교배는 정점의 서열에서만 일어나게 2점 교배를 하며, 교배점은 랜덤하게 선택한다. 돌연변이는 정점의 서열 중 임의의 염기쌍을 선택해 한 염기쌍을 변화시키는 방법을 사용하여 세대수 만큼 반복 한다.

이렇게 하여 높은 적합도를 갖는 우수한 코드를 선택하고 주어진 반응횟수만큼 합성과 분리 과정을 거친다. 이 분리 과정에서 해가 될 가능성이 없는 것은 항체 친화력 반응, PCR(Polymerase Chain Reaction), 겔 전기 영동법 등과 같은 생물학적 연산자를 이용하여 미리 제거한다. 마지막으로 다시 한번 PCR을 이용하여 특정 부위의 서열을 증폭시킨다. 그리고 겔 전기 영동법으로 일정 길이의 염기 배열만을 추출하고, 항체 친화력 반응을 이용하여 그래프의 모든 정점을 한번만 방문한 경로를 선택하여 최종해로 한다.

4. 실험 및 분석

실험은 [그림 1]의 정점 4개와 간선 6개를 대상으로 TSP에 적용하여 ACO와 Adleman의 DNA 컴퓨팅 알고리즘을 비교 평가하였고, [그림 12]의 정점이 6개인 경우도 같이 실험하였다.

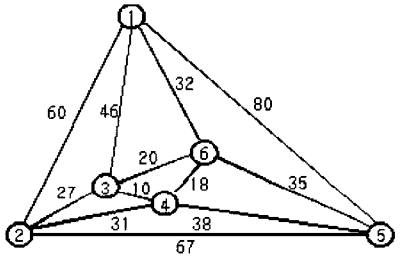


그림 12. TSP 그래프
Fig. 12. Graph of TSP

표 2. 매개변수
Table 2. Parameters

변 수		ACO	Adleman의 DNA 컴퓨팅 알고리즘
집단 크기		1000	1000
세대수		100	100
교배 연산 비율		0.5	0.5
돌연변이 연산 비율		0.1	0.1
임계값		0.1	0.1
총 반응횟수	반복횟수	10	1
	반응횟수	100	1000
생물학 실험 오류확률		0.01	0.01

실험은 P4, 1GHz, RAM 256M의 PC에서 C언어를 사용

하여 평가하였다. 실험에서 사용된 매개변수들은 [표 2]와 같이 설정하였다.

ACO는 합성과 분리 과정을 반복처리 할 수 있으므로 반복횟수를 10, 반응횟수를 100으로하여 이들을 곱한 총 반응횟수는 1000으로 설정하였다. 하지만 Adleman의 DNA 컴퓨팅 알고리즘은 1회의 단순한 합성과 분리 과정을 갖고 있기 때문에 ACO와 동일한 실험 환경을 설정하기 위해 반복횟수는 1, 반응횟수는 1000으로하여 총 반응횟수를 같게하였다. 그리고 DNA 코드의 길이는 Adleman의 DNA 컴퓨팅 알고리즘에서는 최소 10bp, 최대 20bp 사이의 고정길이로 설정하였으며, ACO는 가변길이로 설정하여 실험하였다.

표 3. ACO의 성능
Table 3. Performance of ACO

구분		ACO	Adleman의 DNA 컴퓨팅 알고리즘
평균 적합도	정점4개	0.7032	0.7411
	정점6개	0.7368	0.9095
평균 탐색횟수	정점4개	0.97	0.33
	정점6개	26.98	5.17
탐색시간(s)	정점4개	1.37×104	2.61×10^{4}
	정점6개	2.4×104	4.11×10^{4}

[표 3]과 같이 각 알고리즘에 대하여 평균 적합도, 평균 탐색횟수, 탐색시간을 평가하였다. 그 결과, ACO는 평균 적합도에서 Adleman의 DNA 컴퓨팅 알고리즘보다 더 우수한 적합도를 얻을 수 있었다. 그리고 경로에 대한 탐색 시간은 정점의 길이가 20bp 이상인 ACO의 DNA 코드와 20bp인 Adleman 알고리즘을 비교한 결과 약 반으로 줄어든 것을 확인하였다.

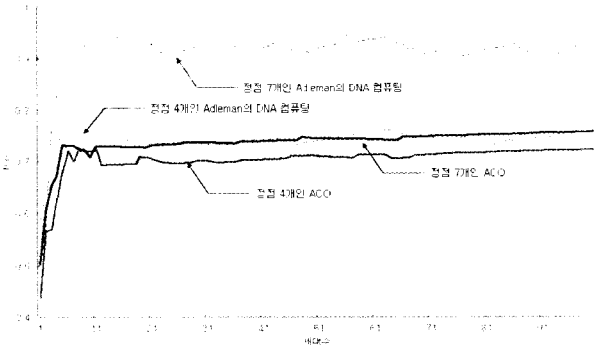


그림 13. 세대별 적합도
Fig. 13. Generation fitness

[그림 13]는 TSP의 세대별 적합도를 나타낸 그래프로 x축은 세대수, y축은 적합도를 나타낸다. 정점이 4개인 경우 ACO는 8세대이후 고른 적합도를 생성하였고, Adleman의 DNA 컴퓨팅 알고리즘은 13세대 이후적합도는 평균 적합도에 가까운 값을 유지하고 있다. 또한 정점이 6개인 경우 ACO의 적합도는 8세대 이후부터 고른 적합도를 유지하고 있지만 Adleman의 DNA 컴퓨팅 알고리즘의 경우 불규칙한 적합도를 생성하였다. 따라서 ACO는 생물학적 실험 오류율을 줄여 높은 적합도를 생성하기 때문에 우수한 해를을 얻을 수가 있었다.

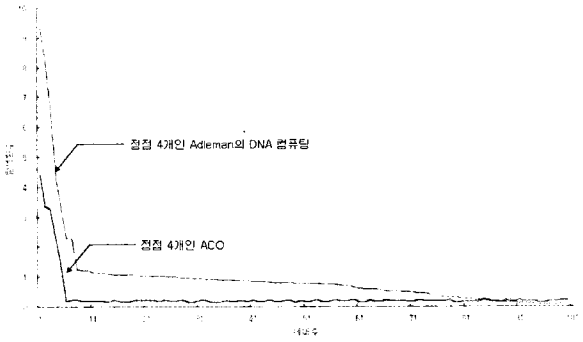


그림 14. 정점 4개의 탐색횟수
Fig. 14. Search number for four nodes

[그림 14]은 정점이 4개인 경우의 탐색횟수를 그래프로 나타낸 것이다. 그래프에서 ACO는 5세대 이후, Adleman의 DNA 컴퓨팅 알고리즘은 8세대 이후에 각각 적합한 경로값을 탐색하였다.

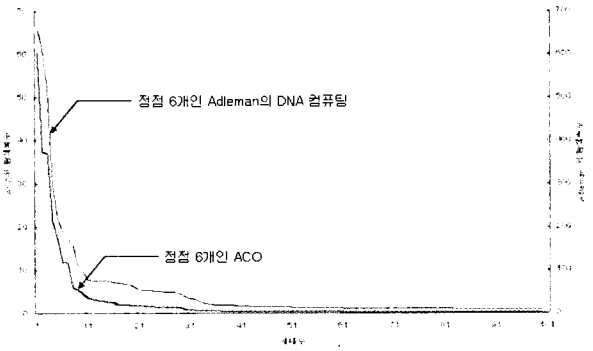


그림 15. 정점 6개의 탐색횟수
Fig. 15. Search number for six nodes

또한, [그림 15]는 정점이 6개인 경우로 ACO는 31세대 이후, Adleman의 DNA 컴퓨팅 알고리즘은 52세대 이후에 각각 우수한 경로 값을 탐색하였다. 이것은 ACO가 생성된 전체 경로 중에서 세대수가 증가함에 따라 많은 수의 부적절한 경로를 짧은 시간 내에 제거하고 적합한 결과를 출력하였다.

[표 4]와 [표 5]는 실험에 사용한 ACO의 가변길이의 정점 코드와 Adleman의 고정길이 정점 코드인 10bp, 20bp에 대한 DNA 코드를 보여주고 있다. [표 6]은 가중치를 포함한 ACO의 간선을 나타냈다.

이 실험에 사용된 [그림 12]에서 최적의 경로는 (V_1, $E_{1>2}$) -> (V_2, $E_{2>3}$) -> (V_3, $E_{3>4}$) -> (V_4, $E_{4>5}$) -> (V_5, $E_{5>6}$) -> (V_6, $E_{6>1}$) -> (V_1, $E_{1>2}$)이며, ACO를 적용한 결과 최적의 서열 코드는 다음과 같다.

(ATGTAGCCCTCATAGGGTGCC,TACAGGCCCGCG) ->
(ATGGCATTTTAGATCCCCGGG, TATATTTATATA) ->
(ATGTTTAACGATTCCAGTAAC, TTTAAT) ->
(ATGTAGCGTTTCCCCGATAGGATCG, TACGACTGG) ->
(ATGCTAGCTTCTAAATAGT, TACGGCTT) ->
(ATGCTAACGGATCTCCCG, TACTTGTTA)

표 4. ACO의 정점에 대한 DNA 코드
Table 4. DNA code for ACO's nodes

정점	ACO의 정점 DNA 코드
1	ATGTAGCCCTCATAGGGTGCC
2	ATGGCATTTTAGATCCCCGGG
3	ATGTTTAACGATTCCAGTAAC
4	ATGTAGCGTTTCCCCGATAGGATCG
5	ATGCTAGCTTCTAAATAGT
6	ATGCTAACGGATCTCCCG

표 5. Adleman의 정점에 대한 DNA 코드
Table 5. DNA code for Adleman's nodes

정점	코드 길이	Adleman의 정점 DNA 코드
1	10bp	ATGCCGGCCT
	20bp	AATCCTATCGCCTTGAACTC
2	10bp	TATTATCGTA
	20bp	CCTAGTCCTATCTGTAACCC
3	10bp	GGTTCCGGTT
	20bp	CGCGAATCCAGCATACTGGT
4	10bp	ATATGCAAGT
	20bp	TCCTAATTGTCCCCTGGTAC
5	10bp	TCCGGAATAT
	20bp	CAACTTTACCTTGTGATCTC
6	10bp	AGTCAGTCCT
	20bp	GATACTAGCCTCTCTAACCC

표 6. ACO의 간선에 대한 DNA 코드
Table 6. DNA code for ACO's edges

간선 (E_i->E_j)	ACO의 간선 DNA 코드		
	Ei	가중치	Ej
1 -> 2	TATCCCACGG	ATGTCCGGGCGC	TACCGTAAAATC
1 -> 3	TATCCCACGG	ATGGCCACATC	TACAAACC
1 -> 5	TATCCCACGG	ATGGGGTTTCCCGCGCG	TACGATC
1 -> 6	TATCCCACGG	ATGATCTT	TACGATT
2 -> 1	TAAAATCTAGGGGCCC	ATGTCAGCGGGCG	TACATC
2 -> 3	TAAAATCTAGGGGCCC	ATATAAATATAT	TACAAACC
2 -> 4	TAAAATCTAGGGGCCC	GGAATTATCTATAT	TACATC
2 -> 5	TAAAATCTAGGGGCCC	ATGACCTACCGTCC	TACGATC
3 -> 1	TAAGGTCATTG	ATGGGCCATC	TACATC
3 -> 2	TAAGGTCATTG	ATTTAAACATAT	TACCGTAAAATC
3 -> 4	TAAGGTCATTG	AAATTA	TACATC
3 -> 6	TAAGGTCATTG	AATTAATTAAG	TACGATT
4 -> 2	TATCCTAGC	ATACTTAGGTCAATA	TACCGTAAAATC
4 -> 3	TATCCTAGC	TTATTA	TACAAACC
4 -> 5	TATCCTAGC	ATGCTGACC	TACGATC
4 -> 6	TATCCTAGC	GAATTTTTTA	TACGATT
5 -> 1	TATCA	ATGCGTTTACCGGCGCG	TACATC
5 -> 2	TATCA	ATGTGCTTAGCGTCG	TACCGTAAAATC
5 -> 4	TATCA	ATGGGAAGC	TACATC
5 -> 6	TATCA	ATGCCGAA	TACGATT
6 -> 1	TAGAGGGC	ATGAACAAT	TACATC
6 -> 3	TAGAGGGC	TTTTATATATC	TACAAACC
6 -> 4	TAGAGGGC	GATAATTTTA	TACATC
6 -> 5	TAGAGGGC	ATGCCGCA	TACGATC

5. 결 론

본 연구에서는 TSP를 이용하여 DNA 염기 서열 디자인에서 발생되는 문제점을 분석하고, 이를 해결하기 위해 DNA 코딩 방법을 적용한 ACO를 제안하였다. 실험에서 ACO는 가변길이의 DNA 코드를 생성하였고, 실제 수소 결합수를 간선의 가중치로 쉽게 표현할 수 있었다. 또한 간선의 가중치 표현 범위를 최대로 확장할 수 있었으며, 짧은 DNA 코드를 가지고도 넓은 범위의 가중치를 표현할 수 있었다. 이렇게 표현된 DNA 코드는 후처리 단계에서 반복적인 합성과 분리 과정을 통해 서열 탐색 시간과 생물학적 오류율이 Adleman의 DNA 컴퓨팅 알고리즘보다 50%정도 줄일 수 있었다.

향후 과제로 더 많은 정점을 갖는 TSP를 NCBI에 등록된 실제 서열을 이용하여 DNA 염기 서열 디자인에 대한 새로운 연구가 필요할 것입니다. 그리고 이렇게 설계된 DNA 염기 서열을 BioInformatics 분야에 적용할 수 있는 가능성을 연구해야 할 것이다.

참고문헌

[1] J. D. Watson, M. Gliman, J. Wikowski, M. Zoller, Recombinant DNA, 2nd Ed., Scientific American Books, New York, 1992.

[2] L. M., Adleman, "Molecular computation of solutions to combinatorial problems", Science, 266:1021-1024, 1994.

[3] A. Narayanan, S. Zorbalas, "DNA algorithms for computing shortest paths", *Genetic Programming 1998*, Koza, J. R. *et al.* (eds.), Morgan Kaufmann, pp. 718-723, 1998.

[4] G. H., Gonnet, C. Korostensky, S. A. Benner, "Evaluation Measures of Multiple Sequence Alignments", Journal of Computational Biology 7(1-2): 261-276 , 2000.

[5] R. Deaton, S. A. Karl, "Introduction to DNA Computing", 1999 Genetic and Evolutionary Computation Conference Tutorial Program, pp. 75-93, Orlando, Florida, July 14, 1999.

[6] T. Yoshikawa, T. Furuhashi, Y. Uchidawa, "Acquisition of Fuzzy Rules of Constructing Intelligent Systems using Genetic Algorithm based on DNA Coding Method" Proceedings of International Joint Conference of CFSA/IFIS/SOFT'95 on Fuzzy Theory and Applications.

[7] T. Yoshikawa, T. Furuhashi, Y. Uchidawa. " The Effect of Combination of DNA Coding Method with Pseudo-Bacterial GA" Proceeding of the 1997 IEEE International Intermag. 97 Magnetics Conference 1997.

[8] O. Martin, S. Otto, E. Felten, "Large-step Markov Chains for the Traveling Salesman Problem.", Complex System, Vol. 5, No. 3, pp. 299-326, 1991.

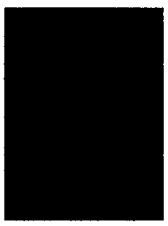
저 자 소 개

김은경(Eun-Gyeong Kim)
2001년 공주대학교 화학과 졸업(학사)
2003년 공주대학교 대학원 컴퓨터공학과 (공학석사)
2003년~현재 공주대학교 대학원 컴퓨터공학과 박사 과정

관심 분야 : 바이오인포메틱스, 인공생명, DNA 컴퓨팅, 유전자알고리즘 등
E-mail : rotnrwk@kongju.ac.kr

윤효근(Hyo-Gun Yun)
1999년 한밭대학교 컴퓨터공학과 졸업 (학사)
2002년 공주대학교 대학원 전자계산학과 (이학석사)
2002년~현재 공주대학교 대학원 컴퓨터공학과 박사 과정

관심 분야 : 인공로봇, 에이전트, 바이오인포메틱스, 유전자알고리즘 등
E-mail : kosher@kongju.ac.kr



이상용(Sang-Yong Lee)
1984년 중앙대학교 전자계산학과 (공학사)
1988년 일본동경대학대학원 총합이공학연구과 (공학석사)
1988년~1989년 일본 NEC 중앙연구소 연구원
1993년 중앙대학교 일반대학원 전자계산학과 (공학박사)
1993년~현재 공주대학교 정보통신공학부 교수
1996년~1997년 University of Central Florida 방문교수

관심 분야 : 인공지능, 에이전트, 컴퓨터게임, 바이오인포매틱스
E-mail : sylee@kongju.ac.kr