해밀톤 경로 문제를 위한 DNA 컴퓨팅에서 코드 최적화

(Code Optimization in DNA Computing for the Hamiltonian Path Problem)

김 은 경 † 이 상 용 ††
(Eun-kyoung Kim) (Sang-yong Lee)

요 약 DNA 컴퓨팅은 생체 분자들의 막대한 병렬성을 정보 처리 기술에 적용한 기술로, NP-complete문제를 해결하기 위하여 사용되고 있다. 하지만 DNA 컴퓨팅 기술만으로 NP-complete 문제를 해결할 경우에는 해를 찾지 못하거나 많은 시간이 걸리는 문제점이 있다.

본 논문에서는 DNA 코딩 방법을 적용하여 DNA 서열을 효율적으로 표현하고, 반응횟수 만큼 합성과 분리 과정을 거쳐 코드를 생성하는 ACO(Algorithm for Code Optimization)를 제안했다. 그리고 ACO를 NP-complete 문제 중의 하나인 Hamiltonian Path Problem에 적용하였다. 그 결과 ACO는 Adleman의 DNA 컴퓨팅 알고리즘 보다 가변길이의 DNA 코드를 효율적으로 표현할 수 있다는 것을 확인하였다. 또한 ACO는 Adleman의 DNA 컴퓨팅 알고리즘 보다 탐색 시간과 생물학적 오류율을 50%정도 줄일 수 있었으며, 빠른 시간 내에 정확한 경로를 탐색할 수 있었다.

키워드 : 해밀톤 경로 문제, DNA 컴퓨팅, DNA 코딩 방법, NP-complete 문제

Abstract DNA computing is technology that applies immense parallel castle of living body molecules into information processing technology, and has used to solve NP-complete problems. However, there are problems which do not look for solutions and take much time when only DNA computing technology solves NP-complete problems.

In this paper we proposed an algorithm called ACO(Algorithm for Code Optimization) that can efficiently express DNA sequence and create good codes through composition and separation processes as many as the numbers of reaction by DNA coding method. Also, we applied ACO to Hamiltonian path problem of NP-complete problems.

As a result, ACO could express DNA codes of variable lengths 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 accurate paths in a short time.

Key words : Hamiltonian path problem, DNA computing, DNA coding method, NP-complete problem

1. 서 론

최근 들어 분자 생물학의 발전으로 생체분자를 이용해 계산을 수행하고자 하는 DNA 컴퓨팅에 관한 연구가 활발해지고 있다. 1994년 Adleman은 DNA가 갖는 막대한 병렬성과 정보 저장능력, Watson-Crick의 상보결합[1]을 이용하여 NP-complete 문제 중의 하나인 Hamiltonian path problem(HPP)을 해결함으로써, 분자 수준의 컴퓨팅이 가능하다는 것을 증명하였다[2]. 뒤이어 1999년에 Ogihara는 DNA가 갖는 화학적인 모델을 Boolean 순회 모델의 분류법에 적용하여 HPP를 해결하였으며[3], 2002년에는 Salomaa가 DNA 컴퓨팅의 특성인 초병렬성을 이용하여 계산 복잡도를 줄이고, Watson-Crick 결합을 재구성한 DNA 염기 서열 생성 모델을 제시하는 등 많은 연구가 이루워졌다[4,5].

DNA 컴퓨팅을 NP-complete 문제 중 HPP에 적용

† 학생회원 : 공주대학교 컴퓨터공학과
rotnrwk@kongju.ac.kr
†† 종신회원 : 공주대학교 정보통신공학부 교수
sylee@kongju.ac.kr
논문접수 : 2003년 5월 9일
심사완료 : 2003년 12월 30일

하였을때 다음과 같은 세 가지 문제점이 발견된다[6]. 첫째, 단순한 합성과 분리 과정을 사용하기 때문에 해를 찾는데 많은 시간과 노력이 요구된다. 둘째, 실제 생물학 실험 방법의 연산자를 사용하므로 실험에 의한 오류의 가능성을 내포하고 있다. 셋째, 그래프 문제를 DNA 코드로 변환할 때 DNA의 특성을 제대로 반영하지 못하고 있다.

현재 이 세가지 문제점을 해결하기 위해 많은 연구가 진행되어 첫 번째 문제는 유전자 알고리즘을 이용한 반복 처리 과정으로 해결되었다[7,8]. 두 번째 문제는 생물학 실험 방법의 메커니즘을 이해하기 위한 연구가 활발히 진행되고 있다[9]. 그러나 세 번째 문제는 아직까지 확실한 방법이 제시되고 있지 않다.

따라서 본 논문에서는 HPP에 DNA 컴퓨팅을 적용하였을때 발생되는 문제점들을 해결하기 위하여 DNA 코딩 방법을 적용한 Algorithm for Code Optimization (ACO)를 제안한다.

2. 관련연구

2.1 Hamiltonian Path Problem

Hamiltonian Path Problem(HPP)는 입력 정점에서 출발하여 모든 정점을 단 한번만 포함한 후, 출력 정점에 도착하는 대표적인 NP-complete 문제 중의 하나이다. 이 문제에 대하여 실제로 주어진 다항식 시간에 해를 구할 수 있는 효율적인 알고리즘은 따로 존재하지 않는다.

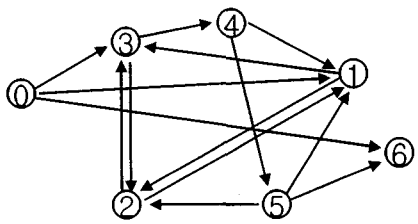

그림 1 Hamiltonian path problem

그림 1은 Adleman이 사용한 HPP의 방향성 그래프로, 정점 0에서 출발하여 모든 정점을 거쳐 정점 6에 도달하는 문제이다.

2.2 DNA 컴퓨팅

DNA 컴퓨팅은 실제 생체 분자인 DNA나 RNA와 같은 살아 있는 세포를 응용한 기술이다. DNA는 이중 나선형 가닥으로 A(Adenine), T(Thymine), C(Cytosine), G(Guanine)라는 4가지 염기로 구성되며, 정해진 규칙에 의해 Watson-Crick 결합을 하고 있다[1]. 그리고 복잡한 염기 조합의 패턴은 하나의 유전 정보를 담고 있으며, 인체내에서 자연 발생하는 효소에 의해 읽혀지고 있다.

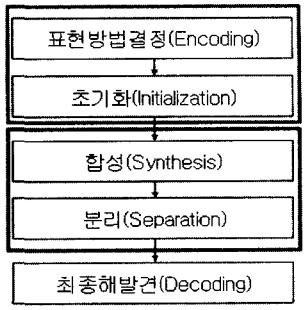

그림 2 Adleman의 DNA 컴퓨팅 알고리즘

그림 2는 Adleman의 DNA 컴퓨팅 알고리즘을 나타낸 것이다. Adleman은 주어진 문제에 맞게 DNA 분자 구조를 표현하고, 생성된 DNA 코드를 단 한번의 합성과 분리 과정을 거쳐 그림 1의 HPP 경로 문제를 해결하였다[2].

이러한 DNA 컴퓨팅의 특징은 매우 낮은 에너지로 작동되며, 나노 수준의 초병렬성을 이용하여 NP-complete 문제에 대해 효과적으로 접근이 가능하다. 또한 계산 속도와 정보의 저장 및 처리 효율에서도 우수함을 보여주고 있다[4,9].

2.3 DNA 코딩 방법

DNA 코딩 방법은 1995년 Yoshikawa가 제안한 변형된 형태의 유전자 알고리즘으로 DNA를 이용한 선택, 재생, 교배, 돌연변이 연산자를 사용한다[10,11]. 또한 DNA 기능을 유지하며, DNA 코드의 길이를 크게 줄일 수 있는 특징을 갖고 있다. DNA 코딩 방법에서는 DNA 코드를 codon(3개의 bases)별로 하나의 아미노산으로 번역한다.

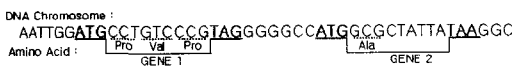

그림 3 DNA 염색체의 번역의 예

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

DNA 코딩 방법의 특징을 살펴보면, 첫 번째로 그림 4에서 보는 것처럼 염색체의 중복을 효율적으로 표현할 수 있다는 것이다. 두 번째로 하나의 아미노산을 만드는 코돈이 여러 개이므로 지식 표현이 쉽다는 것이다. 세 번째로 교차점이 임의로 주어지기 때문에 염색체의 길

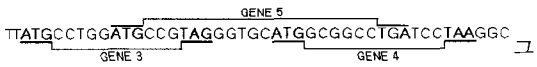

그림 4 유전자 중복의 예

이가 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

HPP를 해결한 Adleman의 DNA 컴퓨팅 알고리즘은 주어진 그래프 문제를 DNA로 변환할 때, DNA의 특성을 제대로 반영하지 못하고 있다. 또한 단 한번의 합성과 분리 과정으로 해를 찾기 때문에 화학적 오류를 포함하고 있다. 따라서 잘못된 해를 찾거나 해를 찾지 못하는 경우가 발생하게 된다.

본 논문에서는 Adleman의 DNA 컴퓨팅의 코딩 문제를 개선하기 위해 DNA 코딩 방법을 적용하여 DNA 코드를 생성하였다. 그리고 DNA 코딩 방법의 연산자를 이용하여 합성과 분리 과정을 반복 처리하므로서 해를 찾는 ACO를 제안한다.

ACO는 HPP 문제에 대하여 DNA의 특징을 최대한 반영한 코드를 생성할 수 있으며, 비효율적인 탐색으로 인한 시간과 노력이 낭비되는 것을 줄일 수 있다. 또한 반복적인 합성과 분리 과정을 통해 생물학적 실험 오류율을 최소화하여 많은 해를 탐색할 수 있다.

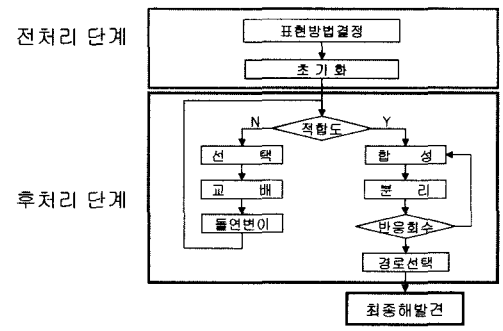

그림 5 ACO의 전체 흐름도

그림 5는 ACO의 전체 흐름도를 표현한 것으로 전처리 단계와 후처리 단계로 구분된다. 먼저 ACO의 전처리단계를 살펴보면, 표현 방법 결정 과정과 초기화 과정으로 나뉜다. 표현 방법 결정 과정에서는 주어진 염기 서열에 대하여 DNA 분자가 가지고 있는 특성을 잘 반영되도록 DNA 코딩 방법을 적용하여 가변길이의 정점과 간선을 생성한다. 정점과 간선은 바로 표현할 수 없으며, 이들을 DNA 염기 서열로 변환하기 위해서는 그림 6과 같은 처리조건을 따른다.

먼저 정점을 생성하기 위하여 모든 DNA 코드에 대해 start codon(ATG)의 위치를 파악하고, i번째 위치와 (i+1)번째 위치 전까지의 사이를 정점으로 표현한다. 단, DNA code의 처음 위치가 start codon으로 시작되지 않을 경우 DNA 코드의 처음 부분부터 i번째 위치 전까지를 정점으로 한다.

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

Begin
  If Middle(DNA_seq, 1, 3) = S_codon 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 > Node_t_No Then Exit for
        V(i) = V(i) + codon
    Next
 End Sub
```

그림 6 정점 표현 프로시저

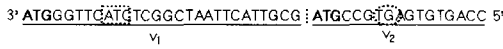

그림 7 정점 표현의 예

그림 7은 그림 6에 따라 V_1과 V_2를 생성한 예이다.

이렇게 표현된 정점을 이어주는 간선은 모든 DNA 코드에 대하여 그림 8에 따라 생성된다. 먼저 점정 Vi에서 처음으로 나타나는 AT*(ATT, ATC, ATA)를 Ei로 지정하고, $V_{(i+1)}$에서 처음으로 나타나는 stop codon인 TAA, TGA, TAG를 $E_{(i+1)}$로 지정하여 두 정점 사이의 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 상보 결합하여 표현한다.

그림 9는 그림 8에 따라 정점 V1과 V2를 이어주는 간선에 대한 표현예이다.

위와 같은 방법으로 정점과 간선을 생성한 후 DNA

```
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) + _
            Middle(V(i+1), 1, Integer(Length(V(i+1))/2))
        Else
          E(i) = E(i) + Middle(V(i+1), 1, En)
        End If
    E(i) = E(i)
   Next
End
```

그림 8 간선 표현 프로시저

5' TAGAGCCGATTAAGTAACGC | TACGGCACT 3'

E_1 E_2

그림 9 간선 표현의 예

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 후처리 단계를 살펴보면 적합도 평가를 하여 적합도를 만족하지 않을 경우, DNA 코딩 방법의 연산자를 이용하여 적합도를 재평가한다. 그리고 만족하는 경우 반응횟수만큼 합성과 분리 과정을 거쳐 우수한 경로를 선택하여 최종해로 한다.

표 1 각 아미노산에 부여된 코드

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

DNA 코딩 방법을 이용하여 그래프 문제에 맞도록 변형된 DNA 코드에 대한 적합도 평가는 표 1과 식 (1)을 이용하여 계산한다. 표 1은 20개의 아미노산 코드를 질량순으로 정렬하여 가중치를 부여하였다. 그리고 식 (1)과 교배, 돌연변이에서 사용되는 연산자들을 DNA 코딩 방법이 유전자 알고리즘의 변형된 형태를 갖기 때문에 유전자 알고리즘의 연산자를 적용하였다. 그래서 식 (1)은 비례 선택법(roulette wheel)으로 상수 k값을 아미노산 코드 값을 적용하였다. 교배는 2점 교배를 갖도록 하고 국소 해에 빠질 위험성을 벗어나기 위해 램덤하게 교배점을 선택하도록 하였다. 그리고 돌연변이는 모든 코드에서 수행하며, 반복은 세대수 만큼 반복하도록 하였다.

$f(x) = x + k|\sin(32x)|, 0 \le x < \pi, k$: 아미노산상수 (1)

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

ACO의 성능을 확인하기 위하여, 앞에서 나온 그림 1의 정점 7개와 간선 14개를 갖는 HPP를 대상으로 ACO, Adleman의 알고리즘을 비교 평가 하였다. 또한 ACO의 각 단계별 성능 평가를 위해 ACO의 전처리단계와 후처리단계를 각각 Adleman의 DNA 컴퓨팅 알고리즘에 적용하여 함께 비교 평가하였다.

표 2 DNA 컴퓨팅에 사용한 매개변수

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
정점표현 코드 길이		가변길이	10bp(최소), 20bp(최대)
총 반응횟수	반복횟수	10	1
	반응횟수	100	1000
생물학 실험 오류확률		0.01	0.01

실험은 PIV, 1GHz, RAM 256M의 PC에서 C언어를 사용하여 진행되었다. 실험에서 사용된 매개변수들은 표 2와 같이 설정하였다. 총 반응횟수에 있어서 ACO와 Adleman의 DNA 컴퓨팅 알고리즘의 값을 동일하게 하기 위하여 반복횟수와 반응횟수를 곱한 1000으로 설정하였다. 이것은 Adleman의 DNA 컴퓨팅 알고리즘이 단 1회의 합성과 분리 과정을 가지고 있으므로 반응횟수를 1000으로 높게 설정하여 ACO와 동일한 총 반응횟수를 갖도록 하였다. 그리고 ACO는 DNA 코드가 가변길이, Adleman의 DNA 컴퓨팅 알고리즘에서는 DNA 코드가 최소 10bp, 최대 20bp 사이의 고정길이로 설정하였다.

표 3 ACO의 비교 평가값

구분	ACO (20bp 이상)	Adleman의 DNA 컴퓨팅 알고리즘 (20bp)	Adleman + ACO 전처리단계 (20bp이상)	Adleman + ACO 후처리단계 (20bp이상)
평균 적합도	0.783	0.464	0.762	0.779
평균 탐색횟수	27	14	12	14
탐색시간(s)	4.6×10^4	7.9×10^4	6.1×10^4	7.3×10^4

표 3과 같이 각 알고리즘에 대하여 평균 적합도, 평균 탐색횟수, 탐색시간을 평가하였다. 그 결과, ACO는 평균 적합도에서 다른 3가지 알고리즘 보다 더 높은 평균 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 알 수 있었다.

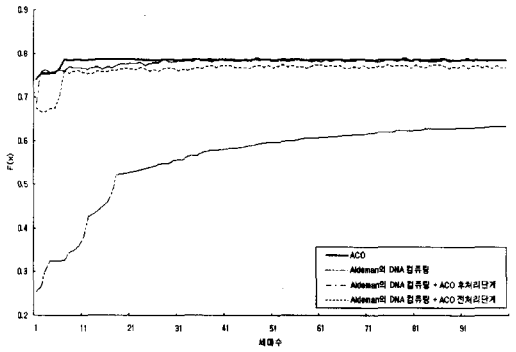

그림 10 세대별 적합도

그림 10은 세대별 적합도를 그래프로 나타낸 것이며 X축은 세대수, Y축은 적합도를 의미한다. 적합도 그래프에서 ACO는 Adleman의 DNA 컴퓨팅 알고리즘 보다 높은 적합도를 보이며, 8세대 이후 고른 적합도 값을 생성하고 있다. 그리고 Adleman의 DNA 컴퓨팅 알고리즘에 적용한 전처리단계와 후처리단계는 각각 높은 적합도를 생성하지만 ACO 보다는 불규칙한 값을 갖는다. 따라서 ACO는 생물학적 실험 오류율을 줄여 높은 적합도를 생성하기 때문에 우수한 해들을 얻을 수가 있었다.

그림 11은 세대별 전체 경로에 대한 탐색 횟수를 나타낸 그래프이다. 그래프에서 x축은 세대수를 나타내며 y축은 전체 경로의 탐색 횟수를 나타낸다. 생성된 전체 경로 중에서 세대수가 증가함에 따라 ACO는 많은 수의 부적절한 경로를 짧은 시간 내에 제거하고 적합한 결과를 출력하였다. 특히 28세대 이후에는 탐색 횟수가 거의 동일함을 알 수 있다.

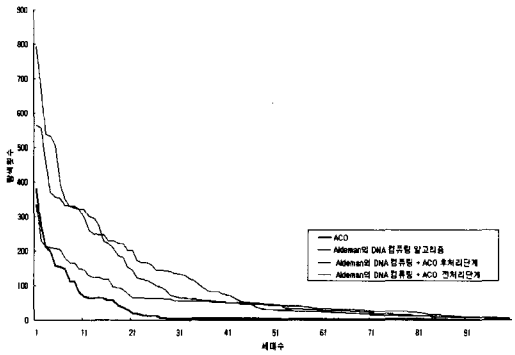

그림 11 세대별 탐색횟수

표 4 DNA 염기 서열 길이에 대한 생물학적 오류율

구분	ACO	Adleman의 DNA 컴퓨팅 알고리즘	Adleman + ACO 전처리단계	Adleman + ACO 후처리단계
오류율	10.3%	21%	22.4%	16.3%

또한 표 4의 생물학적 오류에서도 ACO가 평균 10.3%로, 다른 3가지 알고리즘 보다 오류율이 훨씬 낮다는 것을 확인하였다.

표 5와 표 6은 실험에 사용한 ACO의 가변길이 정점 코드와 Adleman의 고정길이 정점 코드인 10bp, 20bp에 대한 DNA 코드를 보여주고 있다. 표 7과 표 8은 정점으로부터 생성한 ACO와 Adleman의 간선을 DNA 코드로 보여주고 있다.

이 실험에 사용된 그림 1의 최적의 경로는 0 ->1 ->2 -> 3 ->4 ->5 ->6으로, ACO를 적용하여 나타낸 최적의 결과 코드는 다음과 같이 표현된다.

GCTTGGGTAAGGGATCCTAAC ->
ATGTAGCCCTCATAGGGTGCC ->
ATGGCATTTTAATCCCCGGG ->
ATGTTTAACGATTCCAGTAAC ->
ATGTAGCGTTTCCCCGATAGGATCG ->
ATGCTAGCTTCTAAATAGT ->
ATGCTAACGGATCTCCCG

표 5 실험에서 사용된 ACO 정점 DNA 코드

정점	ACO의 정점 DNA 코드
0	GCTTGGGTAAGGGATCCTAAC
1	ATGTAGCCCTCATAGGGTGCC
2	ATGGCATTTTAATCCCCGGG
3	ATGTTTAACGATTCCAGTAAC
4	ATGTAGCGTTTCCCCGATAGGATCG
5	ATGCTAGCTTCTAAATAGT
6	ATGCTAACGGATCTCCCG

표 6 실험에서 사용한 Adleman의 정점 DNA 코드

정점	코드 길이	Adleman의 정점 DNA 코드
0	10bp	GTACCAATGG
	20bp	GTCCTAATTGAGTCCCGCCT
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 7 실험에서 사용한 ACO의 간선 DNA 코드

간선 (V_i->V_j)	ACO의 간선 DNA 코드
0 -> 1	TAGGATTGTACGGTATGATC
0 -> 3	TAGGATTGTACAAATT
0 -> 6	TAGGATTGTACGATT
1 -> 2	TATCCCACGGTACCGTAAAATT
1 -> 3	TATCCCACGGTACAAATT
2 -> 1	TAAAATTAGGGGCCCTACATC
2 -> 3	TAAAATTAGGGGCCCTACAAATT
3 -> 2	TAAGGTCATTGTACCGTAAAATT
3 -> 4	TAAGGTCATTGTACATC
4 -> 1	TATCCTAGCCGAACCCATT
4 -> 5	TATCCTAGCTACGATC
5 -> 1	TATCACGAACCCATTCCCTAG
5 -> 2	TATCATACATCGGGAGTAT
5 -> 6	TATCATACGATT

5. 결론 및 향후 연구과제

본 연구에서는 DNA 컴퓨팅의 문제점인 그래프 문제를 DNA 코드로 변환할 때 DNA의 특성을 제대로 반영하지 못하는 문제를 DNA 코딩 방법으로 해결하고, 반복적인 합성과 분리를 통해 생물학적 실험 오류들을 줄임으로써 시간과 노력을 절감할 수 있는 ACO를 제안했다. ACO를 HPP에 적용한 결과, DNA의 서열을 압축하여 표현함으로써 코드의 길이를 줄일 수 있었다. 그리고 서열 탐색시간과 생물학적 오류율을 Adleman의 DNA 컴퓨팅 알고리즘보다 50%정도 줄일 수 있었다. 이것은 ACO가 DNA의 막대한 병렬성과 상보성을 보다 효과적으로 처리 가능하다는 것을 보여 주었다.

향후 더 많은 정점을 갖는 HPP와 임의의 DNA 염기 서열이 아닌 NCBI에 등록된 DNA 염기 서열에 대한 DNA 컴퓨팅 연구가 필요할 것이다.

표 8 실험에서 사용한 Adleman의 간선 DNA 코드

간선 (V_i->V_j)	코드 길이	Adleman의 간선 DNA 코드
0 -> 1	10bp	TTACCTACGG
	20bp	TCAGGGCGGATTAGGATAGC
0 -> 3	10bp	TTACCCCAAG
	20bp	TCAGGGCGGAGCGCTTAGGT
0 -> 6	10bp	TTACCTCAGT
	20bp	TCAGGGCGGACTATGATCGG
1 -> 2	10bp	CCGGAATAAT
	20bp	GGAACTTGAGGGATCAGGAT
1 -> 3	10bp	CCGGACCAAG
	20bp	GGAACTTGAGGCGCTTAGGT
2 -> 1	10bp	AGCATTACGG
	20bp	AGACATTGGGTTAGGATAGC
2 -> 3	10bp	AGCATCCAAG
	20bp	AGACATTGGGGCGCTTAGGT
3 -> 2	10bp	GCCAAATAAT
	20bp	CGTATGACCAGGATCAGGAT
3 -> 4	10bp	GCCAATATAC
	20bp	CGTATGACCAAGGATTAACA
4 -> 1	10bp	GTTCACATGG
	20bp	GGGGACCATGTTAGGATAGC
4 -> 5	10bp	GTTCAAGGCC
	20bp	GGGGACCATGGTTGAAATGG
5 -> 1	10bp	CAGGACATGG
	20bp	AACACTAGAGTTAGGATAGC
5 -> 2	10bp	CAGGAATAAT
	20bp	AACACTAGAGGGATCAGGAT
5 -> 6	10bp	CAGGATCAGT
	20bp	AACACTAGAGCTATGATCGG

참 고 문 헌

[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] M. Ogihara, A. Ray, Executing Parallel Logical Operations with DNA, In Proceedings of the IEEE Congress on Evolutionary Computation, pp. 972-979, 1999.

[4] A. Salomaa, DNA complementarity and paradigms of computing, Turku Centre for Computer Science TUCS Technical Reports, No.457, 2002.

[5] N. Jonoska, N. C. Seedman (Eds.), "Preliminary Proceedings of 7th International Meeting on DNA Based Computers," University of South Florida, Tampa, FL, June, 10-13, 2001.

[6] J. A. Rose, R. J. Deaton, D. R. Franceschetti, M. Garzon, and S. E. Jr. Stevens, "A Statistical Mechanical Treatment of Error in the Annealing Biostep of DNA Computation," In [GECCO99], pp. 1829-1834.
[7] R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, S. E. Jr. Stevens, "A DNA based implementation of an evolutionary search for good encodings for DNA computation," Proceedings of IEEE Conference on Evolutionary Computation, IEEE Press, pp. 267-271, 1997.
[8] R. Deaton, R. C. Murphy, M. Garzon, D. R. Franceschetti, S. E. Jr. Stevens, "Reliability and efficiency of a DNA-based computation," Physical Review Letters, 82(2) : 417-420, 1998.
[9] M. Yamamoto, J. Yamashita, T. Shiba, T. Hirayama, S. Takya, K. Suzuki, M. Munekata, A. Ohuchi, "A Study on the Hybridiztation Process In DNA Computing," In [DNA-V], pp. 99-108, 1999.
[10] 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, 1995.
[11] 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.
[12] A. Brenneman and A. E. Condon, "Strand Design for Bio-Molecular Computation," Dept. of Computer Science, University of British Columbia, March 22, 2001.

김 은 경

2001년 공주대학교 화학과 졸업(학사). 2003년 공주대학교 대학원 컴퓨터공학과(공학석사). 2003년~현재 공주대학교 대학원 컴퓨터공학과 박사 과정. 관심분야 는 바이오인포매틱스, 인공생명, DNA 컴퓨팅, 유전자알고리즘 등

이 상 용

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. 관심분야는 인공지능, 에이전트, 컴퓨터게임, 바이오인포매틱스