DNA 코드 유전자 알고리즘을 이용한 Sub-Set Sum 문제의 개선

An Improvement of Sub-Set Sum problem using DNA coded Genetic Algorithm

박 찬 량, 이 병 권, 이 상 구
Chan-Ryang Park, Byong-Kwon Lee, Sang-Gu Lee
한남대학교 정보통신·멀티미디어공학부

Abstract

DNA 컴퓨팅 기법은 실제 생체 분자(bio-molecule)를 계산의 도구로 사용하는 새로운 계산 방법으로, 진화 연산과 결합하여 인공지능의 새로운 분야로 부각되고 있다. 그러나, 실제 생체 분자를 계산의 도구로 사용하기 때문에 기존의 컴퓨터에 적용하기 어렵고, 단순히 합성과 분리라는 간단한 방법으로 해를 구하기 때문에 보다 효과적인 알고리즘을 개발하여야 할 필요성이 있다. 따라서, 본 논문에서는 DNA 컴퓨팅 기법을 컴퓨터에 적용하기 위한 방법으로 DNA 컴퓨팅에서의 코드 합성 기법과 유전자 알고리즘을 이용하여 NP-complete 문제중의 하나인 Sub-Set Sum 문제를 해결하여 그 결과를 분석한다. Sub-Set Sum 문제에서 단순 유전자 알고리즘보다 DNA 코드 유전자 알고리즘이 높은 성능을 보인다.

I. 서론

DNA 컴퓨팅은 분자 생물학과 유전 공학의 발달에 힘입어 생체분자를 실제로 사용하여 계산을 수행한다. 분자수준에서의 컴퓨팅 방식은 그 고유의 초고집적도의 정보 저장 능력 및 생화학 반응에 의한 정확하고도 초병렬적인 연산의 특성으로 인하여 세포 수준 위 단계에서의 진화 연산 모델보다 생체 컴퓨터로서의 구현 가능성이 높고 기술적인 파급효과가 훨씬 클 것으로 기대된다. 현재까지도 DNA 분자가 잠재적으로 가지고 있는 막대한 병렬성을 이용해서 NP-complete 문제들을 해결하고자 하는 연구들이 많이 진행되고 있으며, 그 외에도 여러 가지 새로운 응용들이 개발되고 있다.

그러나 아직 실제 생체 분자의 정보처리 원리에 바탕을 둔 새로운 컴퓨터 기술이 개발되지 않고 있다. 또한, DNA 컴퓨팅 기법은 기존의 컴퓨터에 적용하기 어렵고, 단순히 합성과 분리라는 간단한 방법으로 해를 구하기 때문에 보다 효과적인 알고리즘을 개발하여야 할 필요성이 있다. 따라서, 본 논문에서는 DNA 컴퓨팅 기법을 컴퓨터에 적용하기 위한 방법으로 DNA 컴퓨팅에서의 코드 합성 기법과 유전자 알고리즘을 이용한 알고리즘을 제안하고, NP-complete 문제중에 하나인 Sub-Set Sum 문제를 해결하여 그 결과를 분석해 보고자 한다.

II. 본론

2.1 DNA 컴퓨팅의 기본개념

기본적으로 DNA 컴퓨팅 기법은 실제 생체 분자인 DNA을 계산의 도구 및 정보 저장 도구로 사용한다. 따라서 가상의 DNA 컴퓨터는 A(Adenine), C(Cytosine), G(Guanine), T(Thymine), 4가지 염기로 정보를 표현한다.

즉, 2진수를 사용하는 컴퓨터와는 달리 4진수를 사용한다고도 할 수 있다.

자료 구조로는 DNA 구조인 이중가닥(double strand)이나 단일가닥(single strand)을 사용한다. 정보를 저장하고 추출하는 기본 방법은 Watson-Crick 상보 결합이다. 그림 2.1은 DNA의 구조를 나타내고 있다. 그림에 나타나 있는 것처럼 DNA는 3' - 5' 또는 5' - 3' 의 방향성을 가지며, 서로 반대 방향(3' - 5' vs. 5' - 3')끼리 결합을 형성한다.

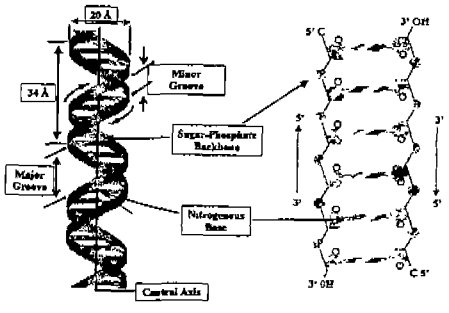

그림 2.1 DNA 구조

DNA 컴퓨팅의 일반적인 연산자로는 생물학 실험 방법들인 결찰법(ligation), 서냉복원법(annealing), 하이브리드형성(hybridization), 중합효소 연쇄반응(Polymerase Chain Reaction, PCR), 겔 전기 영동법(gel electrophoresis), 항체 친화력반응(antibody affinity) 및 여러 가지 효소(enzyme)들을 사용한다[1].

2.2 기본적인 DNA 컴퓨팅 알고리즘

표 2.1 단순화한 DNA 컴퓨팅 알고리즘

1) 표현 방법 결정(Encoding): 분자 구조를 이용하여 주어진 문제에 대한 표현 방법 결정
2) 초기화(Initialization): 결정한 표현 방법을 반영한 분자들을 생성
3) 합성(Synthesis): 결찰법 등의 연산자(실험과정)를 이용하여 가능한 모든 초기 해집합을 생성함
4) 분리(Separation): 생성된 초기해 집합에서 주어진 조건을 만족하지 않는 해들을 분리해 냄
5) 최종해 발견: 분리 과정 이후 존재하는 해들을 최종해라고 판단함.

DNA 컴퓨팅 알고리즘을 간략화 시키면 표 2.1과 같다. 표에서 알 수 있는 것처럼 기본적인 DNA 컴퓨팅 알고리즘은 4단계로 구성된다. 그런데, 합성과 분리 단계는 단순히 생성(generation)하고 검사(test)한다는 과정을 취하고 있기 때문에 개선의 여지가 많이 존재한다. 이러한 기본적인 알고리즘을 개선한 방법을 2.3절에서 소개한다.

2.3 DNA 코드 유전자 알고리즘

유전자 알고리즘을 기반으로 하여 본 논문에서 제안한 문제 표현 방식을 지원하는 Sub-Set Sum 문제 해결을 위한 DNA 코드 유전자 알고리즘을 개발하였다. 알고리즘을 기술하면 표 2.2와 같다. 단계 1에서 3은 초기설정이고, 단계 4는 진화과정이며, 단계 5는 실제로 해를 찾기 위한 과정이다.

표 2.2 DNA 코드 유전자 알고리즘

1 표현 방법 결정(Encoding): 분자 구조를 이용하여 주어진 문제에 대한 표현 방법 결정
2 초기화(Initialization): 결정한 표현 방법을 반영한 코드 군집을 생성
3. 각 코드의 적합도를 계산
4. While (generation $g \leq g_{max}$) do
(a) 새로운 군집을 만들기 위해 유전 연산자를 적용
① 재생(reproduction)
② 교차(crossover)
③ 돌연변이(*mutation*)
[④ 개체군의 크기를 일정하게 유지하기 위한 새로운 염기배열의 추가]
(b) 각 코드의 적합도를 계산
5. 최종 분리
(a) 적합도의 최대치(200)를 갖는 코드 선택

단계 4의 과정에서 최적의 해를 구하기 위해서 유전자 알고리즘을 사용하였다. 각 염기배열들은 각각의 적합도에 비례한 확률적 재생 방법을 사용하였고, 엘리트 전략(elitist strategy)을 적용하였다. 교차 연산은 일점 교차 연산을 사용하였으며, 아래에 나타나 있는 식의 비율로 교차가 일어나도록 하였다.

$$\text{교차 연산 비율} = \frac{(((\text{개체군} - \text{재생개체군})/2) - 1) * 2}{\text{개체군}}$$

앞의 식의 분자 부분에 있는 -1 부분은 새로 생성되는 개체군에서 2 ~ 3 개의 돌연변이가 일어날 수 있는 확률을 갖도록 하기 위해서 사용하였다. 돌연변이 연산은 일정한 확률로 각각의 염기배열에 대하여 하나의 염기쌍이 아닌 일정부분의 가중치 부분을 다른 가중치를 갖는 염기배열로 바꾸어 주도록 하였다. 또한, 돌연변이가 일어나지 않을 경우에 새로운 염기배열을 추가하여 개체군의 크기를 일정하게 유지할 수 있도록 하였다. 이 방법을 통하여 새로운 염기배열을 추가해줌으로써 초기의 염기배열들에 의한 조기 수렴 오류를 막을 수 있을 것이다. 적합도를 계산하기 위해서 부분집합에서 element 크기의 합이 정확히 C가 되는가 하는 정도와 각 element 들이 중복된 정도를 사용하였다.

2.4 Sub-Set Sum 문제

Sub-Set Sum 문제는 크기가 각각 s1, s2, …, sn 인 n개의 element들로 구성된 집합에서 element 크기의 합이 정확히 C가 되는 부분집합이 있는가 하는 문제로 NP-complete 문제로 알려져 있다. Sub-Set Sum 문제는 다음의 집합기호를 사용하여 표현할 수 있다.

Sub-Set sum = $\{ \langle S, C \rangle$: there exists a subset $S' \subseteq S$ such that $C = \sum_{s \in S'} S \}$

2.5 코드 표현 방법

Sub-Set Sum 문제를 풀기 위해서 염기배열의 길이를 가중치에 따라 가변적으로 변화시키지 않고 고정시킨 "고정 길이 표현법(fixed length representation)"을 사용하였다. 염기배열의 길이를 고정시키고, A/T 염기쌍은 0을, C/G 염기쌍은 1을, G/C 염기쌍은 2를, T/A 염기쌍은 3을 각각 나타내는 가중치를 갖도록 표현하였다. 이 경우 결찰법을 수행하기 위해서 간선의 가중치 염기배열 부분만 이중가닥으로 만들고, 위치 염기배열 부분은 단일가닥으로 만들어 점착말단(sticky end)을 구성하도록 한다. 이 방법을 이용하여 실험에서는 표현 영역의 확장을 위하여 가중치 염기배열 부분을 5개의 염기쌍으로 나타내었다. 5개의 염기쌍으로는 0 ~ $4^5 - 1$ 즉, 0 ~ 1023의 범위의 수를 나타낼 수 있다. 그림 2.2에 Sub-Set Sum 문제를 풀기 위해 사용한 염기배열의 예를 나타나낸다.

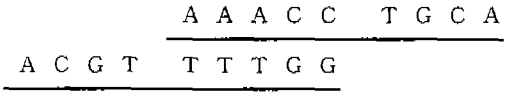

그림 2.2 가중치 5를 갖는 염기배열

이 방법을 사용한 결합과정의 예를 그림 2.3에 보인다.

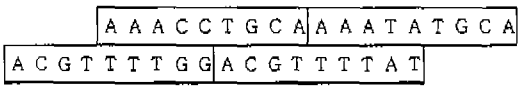

그림 2.3 염기배열 결합 예(원소 5와 10의 결합)

III. 실험 결과 및 분석

3.1 실험 환경 및 변수들

실험에 사용한 집합은 6개의 원소를 가지는 집합이다. 표 3.1은 실험에 사용된 집합을 나타내고 있다.

표 3.1 실험에 사용된 집합

n = 6, C = 30 n(s1, s2, s3, s4, s5, s6) = (5, 10, 12, 13, 15, 18)

프로그래밍에 사용된 여러 변수들은 표 3.2에 기술되어 있다.

표 3.2 진화 연산 실험 조건.

변수	값
개체군 크기	200
세대수	200
재생 비율	적합도에 따른 확률적 재생
교차 연산 비율	$\frac{(((\text{개체군} - \text{재생개체군})/2) - 1) * 2}{\text{개체군}}$
돌연변이 연산 비율	0.05
돌연변이가 생기지 않았을 경우	새로운 염기배열을 추가시켜 개체군의 크기를 일정하게 유지

3.2 실험결과

우선 제안된 방법을 통하여 Sub-Set Sum 문제를 해결할 수 있었다. 진화 과정에서 세대별

염기배열의 총 적합도 변화는 그림 3.1에 나타나 있다. 대략 25세대 정도까지는 적합도가 급격히 증가하지만, 그 이후에는 적합도가 크게 변화하지 않는 모습을 보인다. 즉, 초기에는 적합도가 높은 염기배열들이 다음 세대에도 존재할 가능성이 높기 때문에 전체적인 적합도가 급격히 증가하게 된다. 후반부로 갈수록 적합도의 변화가 거의 없고 일정한 수준에 수렴하고 있음을 볼 수 있다.

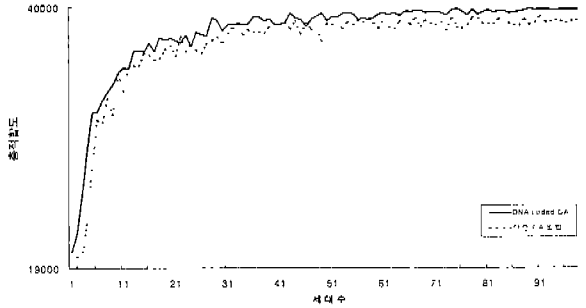

그림 3.1 염기배열 총 적합도 변화 그래프

이 문제에서 적합도는 element 크기의 합이 정확히 C가 되는가 하는 정도와 각 element들이 중복되지 않은 정도를 각각 100으로 나타내므로 200이 최적의 적합도가 된다. 그림 3.2는 각 세대의 염기배열 중에 해를 나타내고 있는 염기배열의 수를 나타내고 있다. 그래프에서 보는 것처럼 해를 나타내는 염기배열의 수가 단조증가하지 않음을 알 수 있다. 이것은 각 염기배열들의 적합도에 비례한 확률적 재생 방법을 사용하였기 때문이다. 즉, 비록 해를 나타내는 염기배열이라도 하나의 해를 나타내는 염기배열들이 계속 중복되는 것을 피하기 위하여 일정한 확률로 재생이 일어나지 않을 수도 있는 것이다.

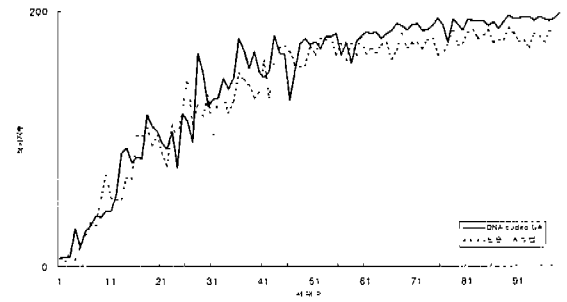

그림 3.2 해를 나타내는
염기배열의 개수 변화 그래프

그림 3.1과 그림 3.2에 나타나 있듯이 DNA 컴퓨팅 기법을 유전자 알고리즘에 적용시킨 방법이 기존의 유전자 알고리즘보다 다소 성능이 향상되었음을 알 수 있다.

IV. 결론 및 향후 연구

본 논문에서는 DNA 컴퓨팅 기술을 인공지능에 적용시키고자 DNA 컴퓨팅 기술과 유전자 알고리즘의 개념을 도입하여 새로운 계산 모델을 제안하였다. 또한, 이를 소프트웨어적으로 구현하고, NP-complete 문제중의 하나로 알려져 있는 Sub-Set Sum 문제를 해결함으로써 기존의 유전자 알고리즘을 개선하였음을 보였다. 본 논문에서 제안된 기법을 사용하면 실제로 사용 가능한 크기의 문제까지 해결할 수 있을 것으로 기대된다.

향후 연구로서는 NP-complete 문제나 최적화 문제에 대한 다양한 응용들의 개발에 관한 연구가 필요하다.

참고 문헌

[1] *영한 미생물학 생화학 면역학 용어집*, 채호석 편저, 유한문화사, 1998.

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

[3] David E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[4] Zbigniew Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, Springer-Verlag, 1992.

[5] Zhang, B. T. and Shin, S. Y., "Molecular Algorithms for Efficient and Reliable DNA Computing", pp.735-742.

[6] Zhang, B. T. and Shin, S. Y., "Code Optimization for DNA Computing of Maximal Cliques", *Advances in Soft Computing - Engineering Design and manufacturing*, Springer-Verlag, 1998.