Smoothing Algorithm을 이용한 DNA 코드 최적화

윤문식°, 한치근
경희대학교 컴퓨터 공학과
mssimsim@algorithms.khu.ac.kr°, cghan@khu.ac.kr

Smoothing Algorithm for DNA Code Optimization

Mun-Sik Yun°, Chi-Geun Han
School of Computer Science, Kyung Hee University

요 약

DNA(Deoxyribo Nucleic Acid)컴퓨팅은 생체분자를 계산의 도구로 이용하는 새로운 계산 방법으로 DNA 정보 저장능력과 DNA의 상보적인 관계를 이용하여 연산을 수행하는 방법이다. 최근에는 DNA 분자들이 갖는 강력한 병렬성을 이용하여 NP-Complete 문제에 적용하는 연구가 많이 시도 되고 있다. Adleman이 DNA 컴퓨팅을 이용해 해결한 HPP(Hamilton Path Problem)와는 달리 TSP(Traveling Salesman Problem)는 간선에 가중치가 추가되었기 때문에 DNA 염기배열로 표현하기가 어렵고 또한 염기배열의 길이를 줄이기 위해 고정길이 염기배열을 사용할 경우 가중치가 커지면 효율적이지 못하다. 본 논문에서는 스무딩 알고리즘(smoothing algorithm)을 사용하여 간선의 가중치를 일정한 비율로 줄인 다음 유전자 알고리즘을 사용하여 최적의 염기배열을 찾는 방법을 제안하였다.

1. 서 론

DNA 컴퓨팅은 실제 생체 분자인 DNA를 계산의 도구 및 정보 저장 도구로 사용한다. 4가지 염기인 A(Adenine,아데닌),C(Cytosin,시토신),G(Guanine,구아닌),T(Thymine,티민)을 사용하여 연산을 수행하는 것이다. 따라서 기존의 컴퓨터에서 사용하는 2진수가 아니라 4진수를 사용한다고 할 수 있다. 또한 DNA 컴퓨팅의 가장 큰 특징은 강력한 병렬성에 있다. DNA 용액 $1ml$ 정도의 액체에 대략 6×10^{20}개 정도의 DNA 분자가 존재하며, 이들을 이용하여 계산하기 때문에 그 병렬성이 매우 크다. 이러한 병렬성은 현재 기술로 해결하지 못하는 NP-Complete 문제등 여러 분야에 적용되고 있다.

DNA 컴퓨팅은 1994년 Adleman이 HPP(Hamilton Path Problem))를 생물학적인 실험 과정만으로 해결함으로써 DNA 컴퓨팅에 새로운 가능성을 제시하였다[1]. 그 이후 Lipton은 Satisfiability(SAT) 문제에 DNA 컴퓨팅을 이용하였고[2], Ouyang은 NP-Complete 문제인 최대 클리크 문제(Maximal Clique Problem)를 해결하였다[3]. 또한 Adleman의 최근 연구에서는 20개의 변수를 포함하고 있는 3-SAT 문제를 DNA 컴퓨팅으로 해결하였다[4]. TSP는 Narayanan과 Zorbalas가 간선에 가중치를 주어 염기배열을 만드는 방법을 제안하였다[5]. 그러나 이 방법은 가중치 편차가 크고 간선의 수가 많아지게 되면 긴 염기배열이 필요하게 되어 매우 비효율적이 된다.

따라서 본 논문에서는 짧은 염기배열의 표현이 가능하도록 스무딩 알고리즘을 사용하였다. 가중치의 스무딩을 통해 일정한 비율을 갖는 값으로 변환한 후 각 가중치의 수소결합수와 비교하여 최적의 염기배열을 찾는 방법을 제안하였다.

본 논문의 구성은 다음과 같다. 2절은 문제정의 및 기존 해법에 대해서 알아보고 3절에서는 제안한 방법을 소개한 후, 4절에서 실험 및 결과를 보이고, 마지막으로 5절에서 결론을 내리고자 한다.

2. 문제 정의 및 기존 해법

2.1 TSP(Traveling Salesman Problem) 문제 정의

무향, 연결 가중 그래프 G=(V, E)가 주어진다. 임의의 노드에서 출발하여 모든 노드를 한번만 방문하고 출발노드로 되돌아오는 경로들 중 가중치의 합이 최소가 되는 경로를 찾는 문제이다.

$$Minimize\ (\sum_{(i,j)\in P} w_{ij}) \qquad [식\ 1]$$

w_{ij}는 노드 i와 j 사이의 가중치이며, P는 경로를 나타낸다.

2.2 기존해법

Narayanan과 Zorbalas가 TSP를 풀기 위해 2가지 방법을 제안하였다[5]. 그 중 개선된 방법에서 가중치 염기배열 표현 하는 부분만 [표 1]을 통해 살펴보겠다.

[표 1] 가중치 표현 방법

1. 가중치를 크기대로 정렬한 후 상수의 곱으로 염기배열의 크기를 결정한다. 가중치의 길이(l_i)= $d_i \times k$ d_i : 가중치의 정렬순서 k: 상수
2. 가중치의 길이만큼 염기배열을 랜덤하게 생성한다.

이 방법은 노드 수와 경로 수가 적을 때는 짧은 염기배열로도 가중치 표현이 가능하지만, 노드 수와 경로 수가 많아지면 매우 긴 염기배열을 표현해야 하기 때문에 비효율적이다. 또한 염기배열이 길수록 결합할 확률이 높기 때문에 큰 가중치를 가지는 배열들이 결합할 확률이 높아 지게 된다.

3 제안한 방법

3.1 스무딩 알고리즘

스무딩 함수는 일정한 비율은 유지한 채 가중치의 값들의 편차를 줄이기 위하여 사용하였다. 전체 간선의 수를 E 라하고 노드 i와 j사이의 거리를 w_{ij}라고 할 때 모든 노드들 사이의 평균거리는 $\bar{w}$가 된다.

$$\bar{w} = \frac{1}{E}\sum_{i \neq j} w_{ij} \qquad \text{[식 2]}$$

$$W_{ij}(\alpha) = \begin{cases} \bar{w} + (w_{ij} - \bar{w})^{1/\alpha} & , \text{if } w_{ij} \geq \bar{w} \\ \bar{w} - (\bar{w} - w_{ij})^{1/\alpha} & , \text{if } w_{ij} < \bar{w} \end{cases} \quad , \alpha \geq 1$$

[식 3]

$$L_{ij} = \frac{1}{w_{ij}} \sum_{i \neq j} W_{ij} \qquad \text{[식 4]}$$

[식 3]의 스무딩 인자(smoothing factor) α가 1인 경우 $w_{ij}(\alpha)$는 원래의 가중치를 가지며, 본 논문에서는 실험을 통하여 적당한 α값을 갖도록 조절하였다. [식 4]에서는 작은 가중치가 높은 수소결합수를 갖기 위해 w_{ij}를 스무딩된 가중치들의 합으로 나누어 실제 가중치와는 반대로 작은 값이 큰 값을 갖게 하였다. 또한 L_{ij} 값들의 차를 이용하여 염기배열의 길이를 결정하였다.

3.2 유전자 알고리즘

유전자 알고리즘을 이용하여 염기배열을 최적화 시켰다. [식 5]는 간선의 가중치 값을 구하는 적합도이며 간선 i의 변환된 가중치(L_i) 와 전체의 의 합($S_{L_{ij}}$), 간선 i의 수소결합수(H_i)와 그래프 전체 가중치의 수소결합수의 합(S_H)으로 각 간선의 가중치 값을 구한다. 최적의 적합도는 0이며 임계값(θ)는 실험을 통해 조절하였다.

$$F_t = \begin{cases} \left| \frac{H_i}{S_H} - \frac{L_{ij}}{S_{L_{ij}}} \right| & \text{if } \left| \frac{H_i}{S_H} - \frac{L_{ij}}{S_{L_{ij}}} \right| \geq \theta \\ 0 & otherwise \end{cases} \qquad \text{[식 5]}$$

비교 대상이 된 유전자 알고리즘은 일점 교차연산(one-cut point crossover)과 일점 돌연변이 연산(one point mutation)을 사용하였으며 적합도 비례 선택(roulette-wheel selection)방법으로 선택 하였다. 자세한 유전자 알고리즘의 변수는 [표 2]와 같다.

4. 실험 및 결과

모두 2가지의 문제에 대해서 실험을 하였다. 비교 대상은 Naraynan과 Zorbalas의 방법으로 가중치 염기배열의 길이를 결정한 것은 [식 6]을 사용하였고, 스무딩한 후 길이를 결정한 것은 [식 5]를 사용하였다. [식 6]은 실제 가중치의 비율에 수소결합수의 비율을 비교한 것이다.

$$F_t = \begin{cases} \left| \frac{H_i}{S_H} - \frac{w_{ij}}{S_{w_{ij}}} \right| & \text{if } \left| \frac{H_i}{S_H} - \frac{w_{ij}}{S_{w_{ij}}} \right| \geq \theta \\ 0 & otherwise \end{cases} \qquad \text{[식 6]}$$

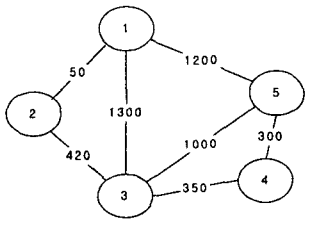
[표 2] 유전자 알고리즘 변수

parameters	값
세대수	1000
개체수	100
돌연변이 연산자 비율	0.2
교차 연산자 비율	0.7
생물학적 오류율	0.01
임계값(θ)	0.001

[표 3] 비교 변수

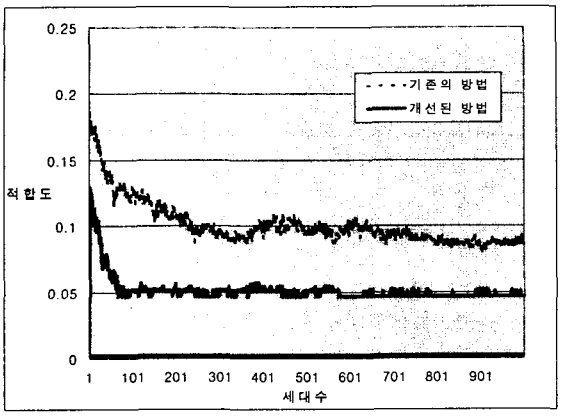
parameters	값
상수(k)	3,4
스무딩 인자(α)	2

[표 3]은 각각의 인자(factor)와 상수들이다. 가중치는 편차가 큰 것과 작은 것 두 가지이며 간선의 개수는 14와 20개로 실험을 하였다. [그림 1]은 14개의 간선을 갖고 가중치의 편차가 큰 그래프이며, 하나의 간선에서 두 개의 경로가 만들어진다. 상수는 편차에 따라 값을 달리하였다. 편차가 큰 14개의 간선을 갖는 그래프에서는 k=4로, 편차가 작은 20개의 간선에는 k=3으로 하였다.

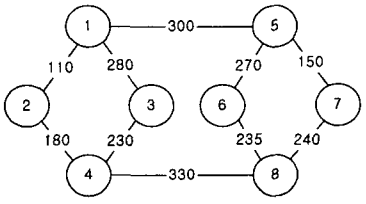


[그림 1] 14개의 간선을 갖는 그래프

[그림 2]는 적합도를 비교한 그래프이다. 제안한 방법은 10번을 실행한 결과 60세대~77세대에서 최적해를 찾았다. 기존의 방법은 200세대 이후에서 최적해를 찾을 수 있었다.

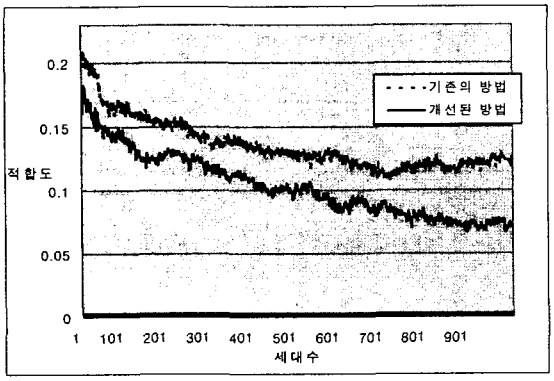


[그림2] 간선 14개의 세대수에 따른 적합도 비교



[그림 3] 20개의 간선을 갖는 그래프

[그림 3]은 20개의 간선을 갖는 그래프이다. α값은 2를 사용하였으며 k 값은 편차가 작기 때문에 3으로 실험하였다. [그림 4]는 적합도를 비교한 그래프이다. 총 20번의 실행에서 기존의 방법은 700세대 이후에서 최적의 염기배열을 찾았으며 개선된 방법으로는 800세대 이후에서 최적의 염기배열을 찾을 수 있었다.



[그림 4]간선 20개의 세대수에 따른 적합도 비교

기존의 방법과 개선된 방법을 비교한 결과를 [표 4]에 나타내었다.

[표 4]기존의 방법과 개선된 방법의 비교

구 분	최적해	최적해 세대수
간선 14개(기존의 방법)	0.0864	200세대이후
간선 20개(기존의 방법)	0.123418	700세대이후
간선 14개(개선된 방법)	0.04701	60~77세대
간선 20개(개선된 방법)	0.07079	800세대 이후

[표 4]에서 알 수 있듯이 스무딩 알고리즘을 사용한 방법이 더 좋은 결과를 보여주었으며, 가중치의 편차에 상관없이 더 좋은 해를 찾았다. 개수가 많은 그래프에서는 제안한 방법이 세대수는 좀더 많아졌지만 적합도가 높은 최적해를 찾을 수 있었다.

5. 결론 및 향후과제

본 논문에서는 간선의 가중치에 염기배열을 적절히 표현하기 위해 가중치를 스무딩함으로써 높은 적합도를 갖도록 하였다. 비교 대상은 가변길이를 갖는 Narayanan과 Zorbalas의 방법 중 가중치 염기배열 생성부분과 비교하였다. 또한 편차가 큰 가중치와 간선의 개수가 많은 가중치로 나누어 실험한 결과, 제안한 방법이 모두 높은 적합도에 최적 염기배열을 찾을 수 있었다. 향후 과제로는 노드 염기배열과의 연결성을 고려함과 동시에 TSP를 풀기위한 합성과 분리 과정을 수행하여야 할 것이다.

참고문헌

[1] Adleman, L. M., "Molecular Computation of Solutions to Combinatorial Problems", Science, vol.266, pp. 1021-1024, 1994.

[2] R,J,Lipton, "DNA Solution of Hard Computational Problems", Science,268:542-545, 1995.

[3] Ouyang, Q., Kaplan, P. D., Liu, S., and Libchaber, A., "DNA Solution of the Maximal Clique Problem", Science, vol.278, pp. 446-449, 1997.

[4] Braich RS, Chelyapov N, Johnson C, Rothemund PW, Adleman L. "Solution of a 20-variable 3-SAT problem on a DNA Computer", Science. 19;296(5567):478-9, 2002.

[5] Narayanan, A. and Zorbalas S., "DNA Algorithms for Computing Shortest Paths", pp.718-724, 1998.

[6] Deaton, R., Murphy, R. C., Garzon, M., Franceschetti, D. R., Stevens, S. E. Jr., "Reliability and Efficiency of a DNA-based Computation", Physical Review Letters, vol.80, no.2, pp 417-420, 1998.

[7] Deaton, R. and Karl, S. A., "Introduction to DNA Computing", 1999 Genetic and Evolutionary Computation Conference Tutorial Program, Orlando, Florida, pp75-93, 1999.

[8] Shin, S. Y., Zhang, B. T., and jun, S. S., "Solving Travelling Salesman Problems Using Molecular Programming", pp. 994-1000, 1999.