Proceedings of KFIS Spring Conference 2007 Vol. 17, No. 1.

대장균의 주화성에 근거한 박테리아 협동 최적화

Bacteria Cooperative Optimization Based on E. Coli Chemotaxis

정희정[1], 정성훈[2]

[1] 서울시 성북구 한성대학교 정보통신공학과
E-mail: davichs@hnsung.ac.kr
[2] 서울시 성북구 한성대학교 정보통신공학과
E-mail: shjung@hansung.ac.kr

요 약

본 논문에서는 박테리아의 주화성에 기초한 Bacteria Cooperative Optimization(BCO) 알고리즘을 소개한다. BCO는 Ant Colony Optimization (ACO)처럼 자연계에 존재하는 생명체의 행동양식을 모방하여 만든 최적화 알고리즘으로 크게 초기화, 측정, 행동결정, 이동으로 구성된다. 우리는 먼저 BCO 알고리즘을 설명하고 2차원 함수 최적화 문제를 이용하여 BCO알고리즘과 Genetic Algorithm(GA) 그리고 Bacterial Foraging for Distributed Optimization(BFO)의 성능 측정 결과를 기술한다. 실험 결과 BCO의 성능이 GA나 BFO보다 우수함을 보였다.

Key Words : Bacteria, Optimization, Chemotaxis, Biomimetics

1. 서 론

최근 생체모방공학(Biomimetics)의 연구와 그 활용이 활발하게 이루어지고 있다. 유전자가 교배나 돌연변이를 하는 것을 이용하여 GA가 개발되었고[4], 개미가 정보를 교환하며 먹이가 있는 곳 까지 가장 최적화된 길을 찾아내는 것에 착안하여 ACO가 개발되었으며[1], 새나 물고기무리가 먹이를 찾아 단체로 이동하며 정보교환을 하는 것을 이용하여 Particle Swarm Optimization(PSO)이 만들어졌다[3]. 이렇게 만들어진 최적화 알고리즘은 각종 공학 문제에 있어서 최적 값을 찾아내는 분야에 많이 응용 되고 있다. 최근에는 생물학에서 유전발현 데이터로부터 유전자 조절 네트워크를 재구성하는 리버스 엔지니어링 분야에서도 많이 사용되고 있다.

ACO는 개미가 실제 군집을 이뤄 행동하는 것을 이용한 최적화 알고리즘이다[1]. ACO의 인조 개미들은 각자 먹이를 탐색하고 그 중 가장 최적화된 길을 따라 먹이를 나르게 된다[2]. PSO에서 각 개체들은 서로 정보를 교환하며 가장 큰 적합도를 갖는 개체 쪽으로 이동하면서 최적 값을 찾는다[3]. GA에서 유전자는 서로 교배하고 돌연변이를 일으켜 최적 값으로 접근하게 된다[4].

BFO[5]와 Bacteria Chemotaxis(BC)[6]는 둘 다 박테리아의 행동양식을 모방하여 만들어진 최적화 알고리즘이다. 박테리아는 지구상의 가장 단순한 단세포 생물로서 박테리아중의 하나인 대장균은 자신에게 유리한 화학물질이 있는 곳으로 이동하고, 불리한 화학물질이 있는 곳은 피하여 움직인다. 대장균의 이러한 행동을 하나의 최적화 과정이라 볼 수 있다[5,6].

본 논문에서 제안하는 BCO는 대장균의 이동 습성에서 착안하여 만든 최적화 알고리즘이다. BCO에서의 대장균은 좋은 화학물질이 있는 곳으로 가고 있다고 판단 할 때는 직선 운동을 하고(run), 반대로 가고 있다고 판단하게 되면 방향을 바꾼다(tumble). 대장균은 화학물질의 농도 변화를 순간적으로 감지하여 판단하는 것이 아니고, 일정 기간 동안 관찰, 기억하여 추이를 지켜본 후 판단한다. 화학물질의 농도가 계속 좋아진다 하여도 일정이상 계속 직선운동 하였다면 무조건 방향을 바꾸는데 이는 더 좋은 곳을 탐색 할 수 있게 진화된 것으로 보인다. 이런 메커니즘을 BCO에 적용함으로써 GA에서 발생하는 조기 수렴(premature convergence) 문제를 완화 시킬 수 있다.

우리는 본 논문에서 제안한 BCO를 포함하여 GA와 BFO를 5개의 함수 최적화 문제에 적용시켜 보았다. 테스트 함수는 비교적 간단한 함수와 여러 논문에서 사용한 복잡한 함수를 이용하였다. 실험 결과 BCO가 GA와 BFO

보다 성능이 우수함을 보였다. 그러나 BCO는 아직 개발 초기 단계로서 성능향상을 위해 향후 보다 많은 연구가 필요하다.

2. BCO 알고리즘

본 논문에서 제안한 BCO 알고리즘은 다음과 같다.

// B_n: 대장균이 직선으로 움직이는 최소 칸 수
// B_m: 대장균이 직선으로 움직일 수 있는 최대 칸 수
// D_n: 최근 먹이 농도를 계산할 칸 수
// D_m: 지난 먹이 농도를 계산할 칸 수
// *run-count*: 직선으로 움직인 칸의 수
// $\bar{\rho}_{D_n}$: D_n칸에서 먹이농도 평균
// $\bar{\rho}_{D_m}$: D_m칸에서 먹이농도 평균

i. 초기화
  a. 초기 N마리의 대장균을 임의의 위치에 생성
  b. 대장균의 초기 방향은 무작위로 설정하고, *run mode*로 설정
ii. 이동 및 측정
  a. 결정 규칙에 의한 먹이농도 측정
  b. 각 대장균마다 행동 규칙적용
  - *run-count*가 B_n과 같으면,
    (a) $\bar{\rho}_{D_n}$이 $\bar{\rho}_{D_m}$보다 크거나 같으면 *run mode*
    (b) $\bar{\rho}_{D_n}$이 $\bar{\rho}_{D_m}$보다 작으면 *tumble mode*
  - *run-count*가 B_m이면, 무조건 *tumble mode*
  - *tumble mode*면, *run-count* 초기화
  c. *run/tumble mode*에 따른 방향으로 대장균 일정 칸 이동하고 *run-count* 증가
  d. 종료 조건 만족 할 때까지 ii.a. 돌아가서 반복
  - 특정 n번 이상 반복 후 종료하거나 각 대장균이 측정한 먹이농도가 특정 값 이상인 경우 종료 (특히 최적 값을 알 때는 최적 값을 찾으면 종료)

초기에 N개의 대장균이 임의의 위치에 생성된다. 각 대장균은 *run mode*나 *tumble mode*를 가질 수 있는데, 초기에는 모두 *run mode*로 설정 한다. *run mode*에 있는 대장균은 B_n칸만큼 움직이고, *tumble mode*에 있는 대장균은 먼저 방향을 전환한 다음 B_n만큼 움직인다. 만약 먹이 농도가 계속적으로 증가하여 대장균이 계속 *run mode*에 있다 하더라도 B_m칸만큼 직선으로 움직인 경우 강제로 tumble 한다. 그러므로 대장균은 최대한 B_m칸만큼 움직인 경우 더 이상 직진 하지 않고 방향을 바꾼다. 이것은 이전 절에서 언급한 것처럼 GA에서 발생하는 조기 수렴 현상을 줄여주는 효과가 있다.

대장균은 D_n칸 동안의 먹이농도 평균($\bar{\rho}_{D_n}$)을 최근 먹이 농도로 사용하고, D_m칸 동안의 먹이 농도 평균($\bar{\rho}_{D_m}$)을 지난 먹이 농도로 사용한다. 만약 최근 먹이 농도가 지난 먹이 농도보다 크거나 같으면 먹이 농도가 증가하는 방향으로 움직이고 있는 것이므로 *run mode*를 유지하고, 그렇지 않으면 *tumble mode*가 된다. 위에서 설명한 것처럼 *run mode*인 경우 같은 방향으로 계속 움직이고 *tumble mode*인 경우 방향을 바꾸어 움직인다. *run-count*는 직선으로 움직인 칸의 수를 저장한다. 그러므로 *tumble mode*인 경우 방향이 바뀌게 되어 *run-count*는 0으로 초기화 된다. 대장균은 종료조건을 만족 할 때 까지 위의 동작을 반복한다. 종료조건은 반복 횟수가 특정 수 이상이 되거나 혹은 어떤 대장균이 최적 값을 찾은 경우가 될 수 있다. 알고리즘에 있는 행동 규칙과 결정 규칙을 요약하면 다음과 같다.

행동 규칙(Behavior Rule): 행동 규칙에서 쓰는 변수는 B_n과 B_m이 있다. 대장균은 B_n칸을 직선 이동한 후에 *run/tumble mode*를 결정한다. 하지만 계속 *run mode*가 되어 직선 이동한 칸 수가 B_m칸이 되었을 때는 무조건 *tumble mode*가 된다($B_n < B_m$).

결정 규칙(Decision Rule): 결정 규칙에서 쓰는 변수는 D_m과 D_n이 있다. 대장균은 한 칸씩 이동할 때마다 자신이 있는 위치의 먹이 농도를 구해 최대 D_m칸 움직일 때 까지 기억한다 ($D_n < D_m$).

3. 실험 및 결과

BCO와 GA및 BFO의 성능을 비교하기 위해 몇 가지의 함수를 이용하여 실험을 하였다. 실험에 사용한 함수는 임의로 만든 simple function과 여러 논문에서 사용한 Mexican hat, schaferI, schaferII, dejongII를 사용하였다. 이 함수들의 최고점이 최적 값을 나타낸다. 각 함수는 다음 식과 같으며 그림 1~5는 각 함수를 그래프로 표현한 것이다.

Simple function:

$$z = 150 - 3(x^2 + y^2)$$

Mexican hat:

$$z = 0.5 - \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1 + 0.001(x^2+y^2))^2}$$

SchaferI:

$$z = 0.5 + (\sin(\frac{(\sqrt{(x^2+y^2)}-0.5)}{(1+0.001(x^2+y^2))^2}))^2$$

SchferII:

$$z = ((x^2+y^2)^{0.25})*(\sin(50*(x^2+y^2)^{0.1})+1)^2$$

DejongII:

$$z = 100(x^2-y)^2 + (1-x)^2$$

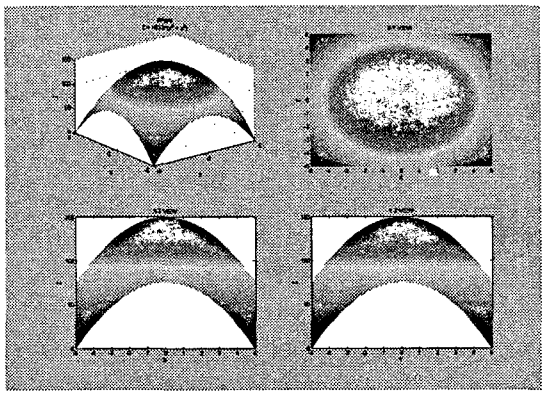

그림 1. Simple function

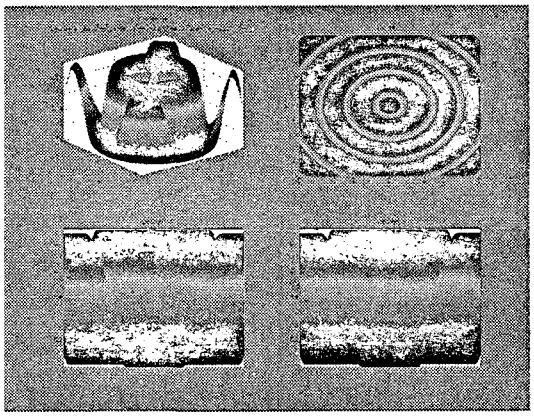

그림 2. Mexican Hat

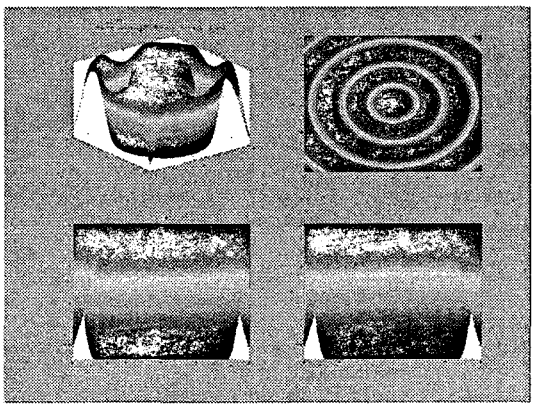

그림 3. SchaferI

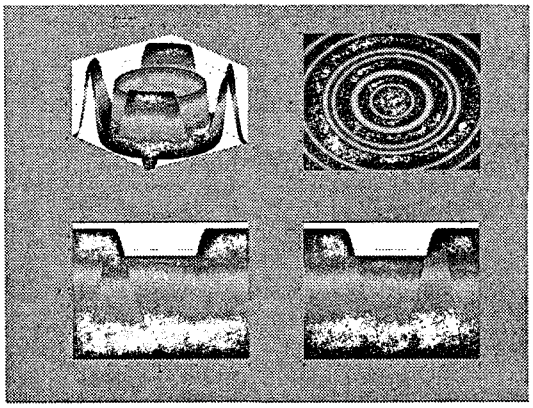

그림 4. SchaferII

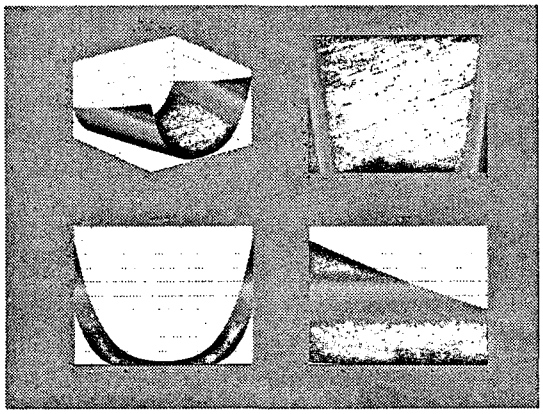

그림 5. DejongII

BCO의 초기 파라미터는 다음과 같다. $D_m = 9$, $D_n = 3$, $B_m = 9$, $B_n = 3$, 대장균의 수 $N = 30$, x축과 y축을 각각 $-5 \sim 5$의 범위를 2^8개로 나누어 실험하였다. 즉, 한번 움직이는 단위는 $\frac{10}{255}$이 된다.

표 1과 2는 BCO와 GA의 성능 비교를 나타낸다. 두 표는 각각 다섯 개의 함수를 50번씩 돌려 나온 50개의 결과의 평균을 낸 결과이다. 표 1은 BCO와 GA가 평균적으로 얼마 만에 최적 값을 찾아내었는가를 나타낸다. BCO는 30마리의 대장균이 모두 한 칸씩 이동하였을 때 수가 하나 증가한다. 예로 BCO의 simple function에서는 30마리가 평균적으로 183.64번 움직여 가장 큰 값을 찾아냈다. 그리고 GA는 30개의 염색체가 모두 교배와 돌연변이를 마치면 수가 하나 증가한다. 표 1에서는 schaferII를 제외한 실험에서 모두 BCO가 높은 성능을 보였다. 표 2는 표 1의 실험에서 나온 BCO와 GA 데이터의 50개 표준편차를 평균 낸 결과이다. schaferII를 제외한 다른 실험에서는 BCO

가 GA보다 비슷하거나 좋은 성능을 보였다.

표 1. BCO와 GA의 성능비교 (평균)

	BCO	GA
simple	183.64	312.74
mexican hat	700.38	1647.4
schaferI	173.30	192.02
schaferII	649.96	365.14
dejongII	1222.30	43389.10

표 2. BCO와 GA의 성능비교 (표준편차)

	BCO	GA
simple	117.80	282.80
mexican hat	1840.14	1791.75
schaferI	124.80	166.14
schaferII	595.66	365.70
dejongII	1137.27	55753.00

BCO와 BFO의 성능 비교 결과는 표 3과 표 4에 있다. 두 표 역시 각각의 실험 함수를 50번 씩 돌려 평균을 낸 결과이다. 하지만 표 3은 표 1과 2의 실험과는 달리 대장균 한 마리가 움직일 때마다 수를 하나 증가 했다. 대략적으로 표 1의 '1'은 표 3의 '30'과 의미가 같다고 할 수 있다. 이렇게 한 이유는 BFO의 알고리즘 특성상 세대별로 횟수를 측정하기 어려웠기 때문이다.

표 3. BCO와 BFO의 성능비교 (평균)

	BCO	BFO
simple	5492.48	23725.52
mexican hat	20996.74	128389.70 (20)
schaferI	5184.84	21231.44
schaferII	19487.58	86855.73 (45)
dejongII	36652.84	50282.93 (14)

표 4. BCO와 BFO의 성능비교 (표준편차)

	BCO	BFO
simple	3536.13	12459.44
mexican hat	55203.14	159899.10 (20)
schaferI	3743.95	35767.98
schaferII	17869.43	119947.50 (45)
dejongII	34118.32	80906.15 (14)

표 4는 표 3의 실험에서 나온 BCO와 BFO 데이터의 30개 표준편차 평균을 구한 값이 나와 있다. 표 3과 표 4의 BFO중 괄호'()'안에 있는 숫자는 50번의 실험 중 결과가 나온 실험의 수이다. 예로 BFO의 Mexican hat은 50번의 실험 중 20번만 결과가 나왔다. 그래서 Mexican hat칸의 평균값은 결과로 나온 20개의 값을 이용해 계산했다. 평균과 표준 편차 둘 다 모든 함수에서 BFO보다 BCO가 좋은 성능을 나타내고 있다. 결과가 나오지 않은 것은 최대 횟수를 반복했음에도 최적 값을 찾지 못한 경우이다. 표에서 보듯이 BFO의 경우에는 최대 횟수를 반복해도 최적 값을 찾지 못하는 경우가 많다. 그러나 BCO에서는 모두 최대 횟수 이전에 최적 값을 찾았다.

4. 결 론

BCO는 박테리아의 주화성을 기초로 한 새로운 최적화 알고리즘이다. BCO와 GA및 BFO의 성능을 비교한 결과 대부분의 실험에서 BCO가 좋은 성능을 보여주었다. 하지만 BCO는 연구 초기 알고리즘으로 수정 및 보완해야 할 곳이 많이 있다. 향후 대장균끼리 상호 작용을 하는 부분과 조기 수렴 문제를 비롯한 여러 부분에 대한 연구를 수행할 예정이다.

참 고 문 헌

[1] Marco Dorigo and Krzysztof Socha "An Introduction to Ant Colony Optimiztion", pp 2-3, 2005

[2] Krzysztof Socha "ACO for Continuous and Mixed-Variable Optimiztion",

[3] James Kennedy and Russel Eberhart "Particle Swarm Optimization", pp 1-2, 1995.

[4] David Beasley and David R. Bull and Ralph R. Martin "An Overview of Genetic Algorithms: Part 1, Fundamentals", 1993

[5] Kevin M. Passino "Biomimicry of Bacterial Foraging - for Distributed Optimization and Control", 2002

[6] Sibylle D. Müller, Jarno Marchetto, Sthfano Airaghi and Petros Koumoutsakos "Optimiztion Based on Bacterial Chemotaxis", 2002