논문 2010-47CI-3-9

성능향상을 위하여 개체속력을 적용한 박테리아 협동 최적화

(Bacteria Cooperative Optimization Applying Individual's Speed for Performance Improvements)

정 성 훈*

(Sung Hoon Jung)

요 약

본 논문에서는 성능향상을 위하여 개체속력 개념을 적용한 박테리아 협동 최적화 방법을 제안한다. 기존의 박테리아 협동 최적화 방법에서는 개체별로 속력이 일정해 모든 개체가 같은 시간에 똑 같은 거리를 움직인다. 이러한 방법은 개체의 적합도가 좋은 개체나 나쁜 개체가 같은 속력으로 움직임으로서 효과적으로 최적 해를 찾아가지 못하는 문제점이 있었다. 이러한 문제점을 개선하고자 개체의 적합도를 이용하여 개체별 등급을 매기고 등급에 따라서 한 번에 이동할 수 있는 거리를 다르게 하는 속력 개념을 적용하였다. 즉 적합도가 낮은 개체는 적합도가 높은 영역으로 빨리 이동하기위하여 속력을 높이고 적합도가 높은 개체는 주변에 최적 해가 있을 가능성이 있으므로 속력을 낮게 유지하였다. 4개의 함수 최적화 문제에 적용해본 결과 속력개념을 적용하지 않은 방법에 대하여 상당한 성능향상이 있음을 보았다. 특히 성능향상을 위하여 기존에 도입했던 등급별 교체방법보다도 더 좋은 성능을 보였다. 이는 박테리아 협동 최적화의 성능을 향상시키기 위한 방법으로 속력개념을 적용하는 것이 매우 유용함을 보여준다.

Abstract

This paper proposes a bacteria cooperative optimization (BCO) method applying individuals's speed for the performance improvements. All individuals in existing BCO methods move the same length at the same time because their speeds are constant. These methods had the problem that the individuals couldn't find the global optimum effectively because good individuals and bad individuals had same speeds. In order to overcome this problem, we applied the speed concept to the BCO algorithm that individuals moved different lengths according to their speeds assigned by the ranks of individuals according to the fitness of individuals. That is to say, we provide high speeds to bad individuals with low fitness in order to fast move to the areas with high fitness and provide low speeds to good individuals with high fitness because they may be near global optimum. It was found from experimental results of four function optimization problems that the proposed method outperformed the existing methods. Our method showed better performances even than the rank replacement method. This means that applying speed concepts to the individuals for BCO is very effective and efficient.

Keywords: Bio-inspired Algorithms, Bacteria Cooperative Optimization, Foraging, Function Optimization

I. 서 론

수 천만 년 간 진화되어온 생물의 여러 가지 특성에서 영감을 얻어 공학적 난제를 해결하려는 시도가 행해져 왔다[1~7]. 대표적인 것으로는 생물의 진화를 모사하여 만든 유전자알고리즘[1~2], 개미가 먹이를 찾아가는 방법을 모사하여 만든 개미군집최적화[5], 새나 곤충의 집단적인 행동으로부터 지적인 행동을 모사하는 집단지능[3~4], 그리고 면역시스템의 특성을 이용하여 만든 인공면역시스템[7] 등이 있다. 이런 알고리즘이 개발된 이후로 많은 곳에 응용되어 왔는데 특히 기존의 방법으로는 한계가 있었던 최적화문제와 지적인 능력을 구현하는데 필요한 탐색 및 진화에 응용되어 왔다.

* 정회원, 한성대학교 정보통신공학과
(Department of Information & Communications Engineering, Hansung University)

※ 본 연구는 2009년도 한성대학교 연구년 지원과제임.

접수일자: 2010년4월5일, 수정완료일: 2010년4월30일

우리는 대표적인 박테리아 중에 하나인 대장균이 먹이를 찾는 행동양식을 연구해왔다[8~10]. 이러한 먹이를 찾기 위한 행동양식을 분석한 결과 이 과정이 하나의 최적화 프로세스로서 이 행동양식을 모사하여 새로운 최적화 알고리즘을 개발할 수 있음을 발견하고 박테리아 협동 최적화 알고리즘을 개발하였다[11~12]. 이 알고리즘에서는 기존의 유전자 알고리즘이나 개체군 최적화와 유사하게 여러 개의 박테리아가 생성되어 탐색 공간 내를 유영하며 먹이가 가장 많은 최적 지점을 찾아나간다[11~12].

첫 번째로 고안한 단순 박테리아 협동 최적화 방법에서는 모든 박테리아가 행동규칙과 결정규칙에 따라 단순히 매 순간 마다 탐색 공간을 한 칸씩 이동하였다[11]. 이러한 방법은 유전자 알고리즘에서 돌연변이 연산자에 의하여 개체가 새로운 영역을 탐색하는 것과 같은 효과를 줄 수 없기 때문에 초기 개체 분포에 성능이 많이 의존하고 지역 최적해에 빠지는 경우가 많아 성능이 좋지 않았다. 이를 해결하기 위하여 제안된 두 번째 방법은 매 세대별 특정 퍼센트의 좋지 않은 개체를 제거하고 무작위적으로 생성된 새로운 개체로 대체하는 방법을 사용하였다[12]. 그 결과 단순 박테리아 협동 최적화 방법보다 성능이 많이 향상되었다. 그러나 새로 대체되는 무작위적으로 생성된 개체가 지역 최적해를 빠져 나오는 데는 도움을 주지만 무작위적으로 생성되기 때문에 찾았던 정보를 모두 잃어버리는 문제점이 있으며 이는 결국 성능을 저하시키는 요인이 된다. 또한 몇 퍼센트의 개체를 대체해야 성능이 좋아지는 가를 일일이 실험해 보거나 경험적으로 선택해 주어야하는 문제점이 있다.

본 논문에서는 성능 향상을 위하여 개체에 속력개념을 적용한 박테리아 협동 최적화 알고리즘을 제안한다. 제안한 방법에서는 개체의 적합도를 이용하여 개체에 등급을 매기고 등급에 따라서 한 번에 움직일 수 있는 속력을 할당한다. 일반적으로 개체의 적합도가 낮은 경우 해당 개체는 전역 최적해로부터 멀리 떨어졌다고 볼 수 있다. 그러므로 개체의 등급이 낮은 경우 속력을 크게 주어 한 번에 많은 거리를 움직일 수 있게 한다. 반면에 개체의 등급이 높은 경우 전역 최적해 근방에 있다고 볼 수 있으므로 속력을 낮게 주어 근방을 탐색하게 한다. 다만 이러한 경향성은 모든 문제에서 반드시 성립하는 것은 아니기 때문에 반드시 좋은 것만은 아니나 확률적으로 가능성이 높기 때문에 성능을 향상 시켜줄 수 있다. 이러한 등급별 속력조절은 좋지 않은 개체를 무작위로 생성된 새로운 개체로 대체하는 방법과 같이 지역 최적해에 빠지는 확률을 줄여주고 비록 지역 최적해에 빠졌다하더라도 빠져나올 수 있는 확률을 높여준다. 더불어 무작위적으로 만들어진 것이 아니라 많은 이동을 통하여 이런 효과를 주기 때문에 무작위 적인 방법보다 기존에 찾은 정보를 덜 잃어버리게 되는 장점이 있다.

본 논문에서 제안한 방법의 성능을 측정하고자 4개의 함수최적화 문제에 적용하여 실험하였다. 실험은 단순 박테리아 협동 최적화 방법과 좋지 않은 개체의 특정 퍼센트를 무작위적으로 만들어진 개체로 대체하는 박테리아 협동 최적화 방법과 더불어 기존에 많이 사용되는 유전자 알고리즘을 이용하여 실험하였다. 실험결과 본 논문에서 제안한 방법이 기존의 유전자 알고리즘은 물론 이전에 제안한 등급별 무작위 개체로 대체하는 방법보다 더 성능이 좋음을 확인하였다. 이로서 제안한 방법이 매우 효과적이고 효율적인 방법임을 확인하였다.

Ⅱ. 박테리아 협동 최적화

1. 대장균의 행동양식 모델링

대표적인 박테리아 중에 하나인 대장균은 자신이 좋아하는 화학물질의 농도가 높아지는 경우 몸통에 붙어있는 나선형의 편모를 시계방향으로 회전하여 전진함으로서 해당 화학물질 쪽으로 이동하여 흡수한다. 반면에 좋아하는 화학물질의 농도가 약해지거나 해를 끼치는 화학물질의 농도가 높아지는 경우 편모를 반시계 방향으로 회전하여 방향을 변화시킴으로서 이를 피한다. 이 과정에서 대장균은 매 시간 스텝별로 화학물질의 농도를 측정하고 이를 기반으로 움직임을 결정 한다[8]. 이를 대장균의 주화성 (chemotaxis)이라 부른다. 우리는 이전의 두 논문[9~10]에서 이러한 대장균의 주화성이 두 개의 단순한 규칙으로 모사될 수 있음을 보였으며 이를 기반으로 단순 박테리아 협동 최적화 알고리즘을 개발하였다[11]. 또한 이러한 박테리아 협동 최적화 알고리즘에 등급별 대체 기법을 적용하여 성능이 향상됨을 보였다[12].

박테리아 협동 최적화를 적용하기 위해서 먼저 최적화 문제는 해당 문제의 차원으로 나뉜 탐색공간을 일정한 개수의 공간으로 나눈 다음 일정 개수의 인공 대장

균을 무작위적으로 분포시킨다. 이 공간에는 최적화 문제에 따라서 주어진 적합도가 할당된다. 이 공간상에 인공 대장균이 행동규칙과 결정규칙으로 이동하면서 최적 적합도를 갖는 위치를 찾게 된다. 행동규칙은 대장균의 행동양식을 분석하여 만든 규칙으로 다음 두 개의 항목으로 이루어져 있다.

(B1) 모든 인공 대장균은 매 B_n 스텝마다 직진할지 방향전환을 할지 결정한다.

(B2) 만약 직진 총수가 $B_m(>B_n)$ 이 되면 무조건 방향전환을 한다.

첫 번째 행동규칙은 인공대장균이 이동 방향 전환 결정을 할 얼마나 자주할 것인지 나타낸다. 결정에 따라서 현재의 방향으로 계속 직진할 수도 있고 다른 방향으로 방향 전환을 할 수도 있다. 방향전환으로 결정된 후에는 방향을 정해야하는데 현재는 2차원일 경우 전체 8방향 중 현재의 방향과 반대 방향을 제외한 6방향 중에 하나를 선택한다. 파라미터 B_n이 작으면 자주 방향결정을 하게 되어 현재의 적합도 공간에 따라서 매우 민감하게 반응하게 되며 B_n 이 크면 덜 민감하게 반응한다. 그러므로 적합도 문제에 따라서 적절한 B_n 값의 선택이 중요하다. 두 번째 행동규칙은 한 방향으로 계속 이동한 개수가 B_m 이 되면 무조건 방향을 전환하게 한다. 여기서 B_m은 당연히 B_n 보다 커야한다. 두 번째 규칙도 실제 대장균의 행동양식에서 따온 것으로 진화적으로 이것이 유리함을 말해준다. 즉 아무리 계속적으로 좋은 화학물질의 농도가 높아진다 하더라도 일정 개수를 이동한 후에는 방향전환을 하는 것이 좋다는 것이다. 이는 최적화 입장에서 보면 지역 최적해에 빠지는 것을 방지해 주는 역할을 한다. 만약 두 번째 규칙이 없으면 대부분의 인공 대장균이 빠르게 지역 최적해에 도달하게 되어 유전자 알고리즘의 조숙수렴현상과 유사한 현상을 발생시킬 것이다. 그러나 B_m 은 지역 최적해에 빠지는 것을 일정부분 완화시켜줄 수 있지만 근본적으로 막지는 못하며 B_m이 너무 작으면 무작위 탐색 경향이 나타난다.

결정규칙은 인공 대장균의 행동을 결정하는 것으로 다음과 같이 두 항목으로 이루어져 있다.

(D1) 인공 대장균은 현재 및 과거의 좋은 화학물질의 농도를 측정하기 위하여 현재로부터 각각 D_n, $D_m(>D_n)$ 시간 동안의 농도를 평균한다.

(D2) 만약, 현재의 화학물질의 평균농도가 과거의 화학물질의 평균농도보다 크면 직진으로 행동을 결정하고 그렇지 않으면 방향전환으로 행동을 결정한다.

첫 번째 결정규칙은 인공대장균이 현재와 과거의 화학물질의 농도를 결정할 때 몇 스텝 동안의 값으로 결정할 것인지를 말해준다. 즉 $D_n=1$이고 $D_m=2$ 라면 현재 위치에서의 농도가 현재의 화학물질의 농도이고 현재 위치에서의 농도와 이전 시간에서의 위치에서의 농도의 평균이 과거의 화학물질의 농도가 된다. 현재와 과거의 농도를 계산하기 위하여 이와 같이 시간이 겹치는 방법을 사용한 것이 겹치지 않는 방법을 사용한 것보다 실험적으로 성능이 더 좋았다. 다만 이는 이론적으로 확정된 결과가 아니기 때문에 문제에 따라서 겹치지 않는 방식이 좋을 수도 있을 것이다. D_n과 D_m은 적합도에 대한 민감도로서 D_n 과 D_m 이 작으면 민감하게 반응하고 크면 덜 민감하게 반응하게 된다. 두 번째 규칙은 방향설정을 위한 규칙으로 농도가 커지는 쪽으로 진행하고 있으면 직진하고 그렇지 않으면 방향을 전환한다는 의미이다. 이는 경사도를 이용하는 기 개발된 대부분의 방법과 유사한 것으로 시간이 지날수록 개체가 더 좋은 곳으로 갈 수 있게 해주는 근거가 된다. 다만 이전에서도 언급한 것처럼 이 규칙이 강하게 적용되면 기존의 언덕 오르기 (hill climbing) 에서와 같이 지역 최적화에 빠지는 문제점이 대두된다. 이를 피하기 위하여 적절한 B_n, B_m 값과 D_n, D_m값을 선정해야하며 이것으로 부족하기 때문에 추가적인 동작이 고안되어야 한다.

2. 박테리아 협동 최적화 알고리즘

1절에서의 모델링을 근거로 하여 알고리즘 1과 같은 박테리아 협동 최적화 알고리즘이 개발되었다. 알고리즘 1에서 ◀ 표시 줄을 제외하면 단순 박테리아 협동 최적화 알고리즘이다. ◀ 표시 줄은 본 논문에서 제안한 속력개념을 적용한 박테리아 협동 최적화 알고리즘에 추가되는 부분이다. 먼저 단순 박테리아 협동 최적화 알고리즘을 설명하면 다음과 같다.

단순 박테리아 협동 최적화 알고리즘은 크게 초기화

알고리즘 1. 개체 속력을 적용한 박테리아 협동 최적화

```
// t : 이산 시간 //
// r_c : 직진 이동 개수 //
// B_n : 인공 대장균이 행동을 결정할 스텝 수 //
// B_m : 인공 대장균이 무조건 방향 전환할 스텝 수 //
// D_n : 현재의 화학물질을 측정하기 위한 스텝 수//
// D_m : 과거의 화학물질을 측정하기 위한 스텝 수 //
// ρ_Dn : 현재의 화학물질 농도 //
// ρ_Dm : 과거의 화학물질 농도 //
// s : 한 번에 이동하는 거리 (속력 적용) // ◀
// E(t) : 시간 t에서의 인공 대장균 집합 //
1  t=0
2  초기화 E(t)
3      탐색 공간 상 에 일정한 분포로 인공 대장균을 무작위적으로 생성
4      생성된 인공 대장균의 방향을 무작위로 결정
5      생성된 인공 대장균의 행동을 직진으로 결정
6      모든 인공 대장균의 직진 이동 개수 r_c, 현재 화학물질 농도 ρ_Dn, 과거 화학물질 농도 ρ_Dm 를 0으로
7      현재의 위치에서 화학물질의 농도를 측정하여 저장
8  while (not 종료조건)
9  do
10     t=t+1
11     이동 E(t)
12         설정된 방향으로 s 만큼 직진 이동
13         직진 이동 수 r_c를 하나 증가
14     측정 E(t)
15         현재의 위치에서 화학물질의 농도를 측정하여 저장
16         ρ_Dn 과 ρ_Dm를 계산              ▷ 결정규칙 (D1)
17     결정 E(t)
18         if (r_c 나머지 B_n)=0  then      ▷ 행동규칙 (B1)
19             if ρ_Dn > ρ_Dm then          ▷ 결정규칙 (D2)
20                 직진 설정
21             else
22                 방향전환 설정
23             end if
24         end if
25         if r_c = B_m          then                ▷ 행동규칙 (B2)
26            방향전환 설정
27         end if
28         if 방향전환 then
29             현재방향과 반대방향을 제외하고 임의의 방향으로 방향 설정
30             직진설정
31             직진 이동 개수 설정, r_c=0
32         end if
33     속력결정 E(t)    ◀
34         E(t)를 현재 위치의 화학물질의 농도로 정렬  ◀
35         화학물질의 농도가 높은 것을 1등으로 하여 등급을 설정 ◀
36         등급을 해당 개체의 이동 거리 s로 설정 ◀
37 end
```

와 이동, 측정, 결정의 단계로 구성되어 있다. 초기화는 특정 개수의 인공 대장균을 탐색 공간에 일정 분포로 무작위적으로 생성하는 단계이다. 방향은 무작위적으로 설정하며 행동은 직진으로 설정한다. 처음 만들어 졌기

때문에 모든 값은 0으로 초기화하고 현재 위치에서의 화학물질의 농도를 측정하여 저장한다. 이동은 인공 대장균이 현재 위치에서 결정된 방향으로 한 칸 이동하는 행동이다. 직진 이동 수 r_c를 하나 증가 시킨다. 측정은 현재 위치에서의 화학물질의 농도를 측정하여 저장하는 것이다. 측정된 화학물질의 농도를 이용하여 현재 및 과거의 화학물질의 농도를 계산한다. 만약 현재까지 진행된 스텝이 농도를 계산하는데 필요한 스텝보다 작은 경우 이용 가능한 스텝만으로 계산한다. 결정은 행동을 결정하는 단계이다. 행동 결정은 직진 이동 수 r_c 가 행동 결정 스텝 수 B_n만큼 이동할 때마다 결정한다. 즉 r_c를 B_n으로 나눈 나머지가 0이 될 때마다 결정한다. 예를 들어 B_n이 3이면 직진 이동 개수가 3, 6, 9,...마다 새롭게 행동을 결정한다. 행동을 결정하는 것으로 결정되면 현재 화학물질의 농도 ρ_{D_n} 과 과거 화학물질의 농도 ρ_{D_m}를 비교하여 현재가 크면 직진, 과거가 크면 방향전환으로 결정한다. 두 번째 행동결정 규칙에 의하여 r_c가 B_m이 되면 무조건 방향전환 한다. 방향전환 하는 것으로 결정되면 현재방향과 반대방향을 제외하고 무작위적으로 방향을 결정한 다음 직진으로 설정하고 직진 이동 개수를 0으로 한다. 이와 같은 동작을 반복하면서 인공 대장균은 대상 최적화 문제에 따라 설정된 적합도 공간을 이동하면서 전역 최적해를 탐색하게 된다. 보다 자세한 설명은 논문[11]을 참조하기 바란다.

이전 절에서도 이야기 했듯이 박테리아 협동 최적화는 두 번째 행동규칙 (B2)와 첫 번째 결정규칙 (D1)에 의하여 언덕 오르기와 같은 일반적인 경사에 근거한 알고리즘 보다는 지역 최적해에 빠지는 문제가 많이 발생하지 않는다. 그러나 B_n, B_m, D_n, D_m이 일정한 값으로 결정된 상황에서는 지역 최적해에 빠지는 문제는 발생할 수

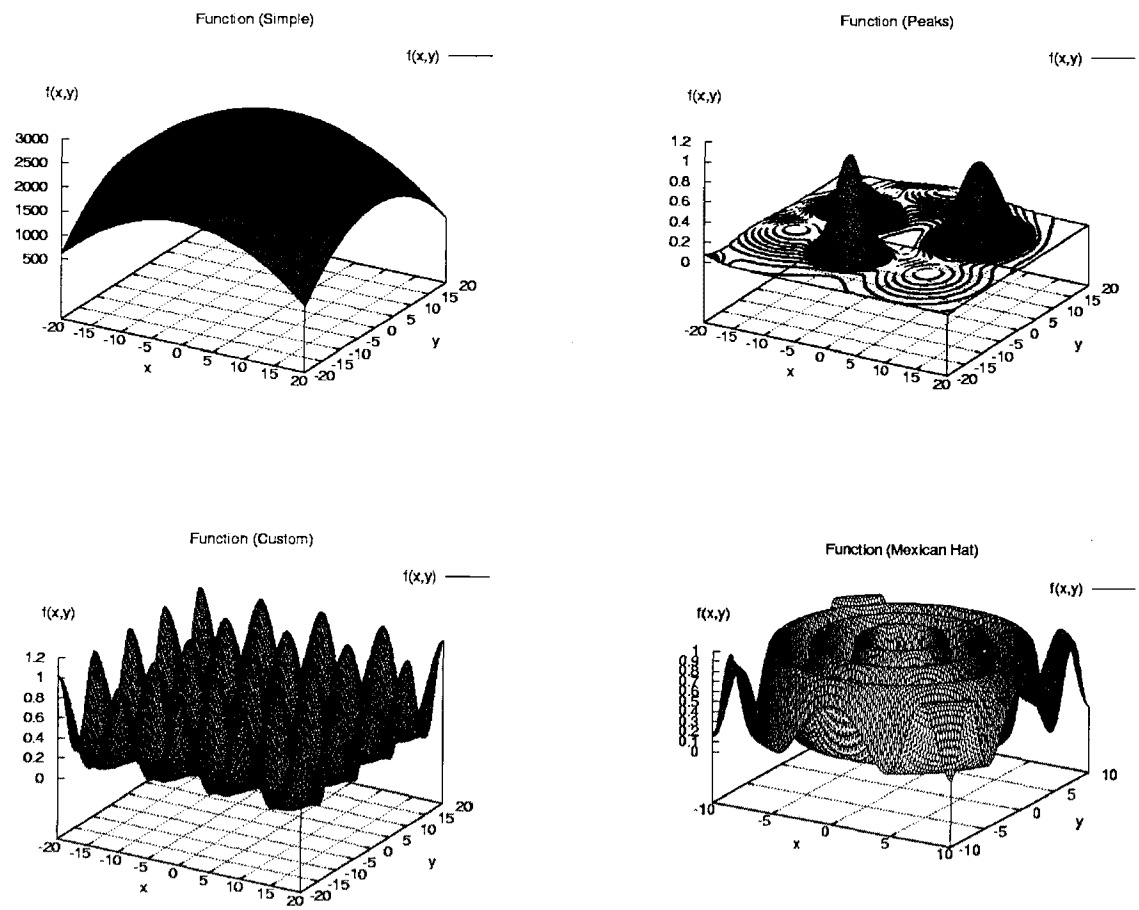

그림 1. 함수 최적화 문제 (a) f_1 (b) f_2 (c) f_3 (d) f_4
Fig. 1. Function optimization problems (a) f_1 (b) f_2 (c) f_3 (d) f_4.

있으며 이러한 문제가 박테리아 협동 최적화 알고리즘의 성능을 저하시킨다. 유전자 알고리즘의 조숙수렴현상처럼 이러한 문제를 해결하지 않고는 만족할만한 성능을 기대하기 어렵다. 이러한 관찰을 토대로 적합도가 낮은 일정 퍼센트의 개체를 무작위적으로 생성된 개체로 대체시키는 방법을 제안하였으며 일정부분 성능 개선이 있었다[12]. 그러나 무작위적으로 생성된 개체는 임의의 위치에 생성되기 때문에 지역 최적해를 빠져 나오는 데는 도움을 주나 탐색적 측면에서는 비효율적인 면이 많은 방법이다.

본 논문에서는 개체의 적합도를 이용하여 등급을 매기고 등급에 따라서 한 번에 이동할 수 있는 거리를 할당하여 좋은 개체는 근처를 샅샅이 뒤지고 나쁜 개체는 멀리 이동하게 하여 전역 최적해를 세부적으로 찾으면서도 지역 최적해에 빠지지 않거나 빠졌을 때는 쉽게 빠져나오는 방법을 제안한다. 알고리즘 1에서 33~36 줄이 속력개념을 적용하기위해 기존의 단순 박테리아 협동 최적화 알고리즘에 추가된 부분이다. 해당 줄에서 보듯이 현재 위치에서 화학물질의 농도를 이용하여 화학물질의 농도가 높은 것을 1등으로 등급을 매기고 속력을 등급 값으로 설정한다. 즉 화학물질의 농도가 가장 높은 것은 1칸씩 이동하며 등급이 낮은 것은 해당 등급 수만큼 이동하게 된다. 예를 들어 20개의 개체가 있다고 하면 1등부터 20등까지 등급이 매겨지고 1등은 1칸씩 20등은 한번에 20칸씩 이동한다. 이렇게 함으로서 등급에 따라서 한 스텝에 움직이는 거리, 즉 속력이 할당된다. 결국 높은 등급은 천천히 움직이며 낮은 등급은 빠르게 움직이게 된다.

III. 실 험

본 논문에서 제안한 방법의 성능을 측정하고자 4개의 함수 최적화 문제에 적용시켜 실험하였다. 사용한 4개의 함수 최적화 문제는 수식 (1) 과 같다. 함수 f_3의 파라미터는 표 1과 같다. 4개의 함수를 그래프로 표현하면 그림 1과 같다. 함수 f_1은 단순한 함수로서 하나의 지역해가 바로 전역해 이므로 비교적 쉬운 최적화 문제이다. 이에 비하여 함수 f_2는 전역 최적해 옆에 크고 작은 지역해 여러 개를 갖는 함수이다. 함수 f_3는 최적해와 매우 유사한 값을 갖는 다수의 지역 최적해를 전체 탐색 영역에 골고루 갖는 문제로서 지역 최적해에 빠지기가 매우 쉬운 문제이다. 표 1이서 보듯이 전역 최적

표 1. 함수 f_3의 파라미터
Table 1. Parameters of function f_3.

i	1	2	3	4	5	6	7	8	9
A	1.00	0.99	0.98	0.99	0.98	0.98	0.99	0.99	0.99
B	0	0	0	0	0	10	10	10	10
C	3	3	3	3	3	3	3	3	3
D	0	-10	-20	10	20	20	10	0	-10
E	3	3	3	3	3	3	3	3	3
i	10	11	12	13	14	15	16	17	18
A	0.98	0.98	0.99	0.99	0.99	0.98	0.98	0.99	0.99
B	10	20	20	20	20	20	10	10	10
C	3	3	3	3	3	3	3	3	3
D	-20	20	10	0	-10	-20	20	10	0
E	3	3	3	3	3	3	3	3	3
i	19	20	21	22	23	24	25	26	27
A	0.99	0.98	0.98	0.99	0.99	0.99	0.98	0.04	0.04
B	10	10	20	20	20	20	20	0	0
C	3	3	3	3	3	3	3	30	30
D	-10	-20	20	10	0	-10	-20	0	0
E	3	3	3	3	3	3	3	30	30

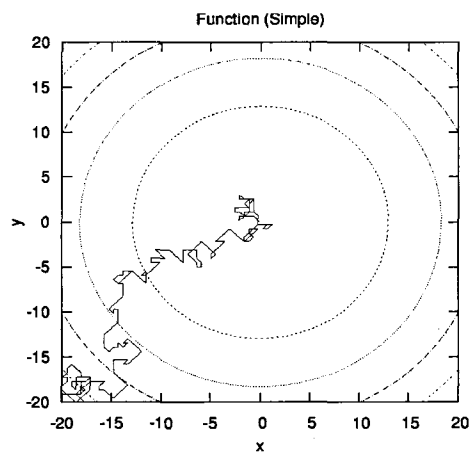

그림 2. 함수 f_1에서의 인공 대장균의 이동 추적
Fig. 2. Moving trajectory of artificial E. Coli at function f_1.

해는 크기 1을 갖는 첫 번째에만 존재하며 다른 영역은 모두 지역 최적해에 해당한다. 마지막으로 함수 f_4는 멕시코 사람들이 쓰는 모자와 같이 생겼다고 해서 Mexican hat 로 불리는 함수이다. 이 함수는 정 가운데 1을 갖는 전역해를 제외하고는 주변에 원형의 형태로 일정거리를 두고 수많은 지역 최적해가 퍼져있는 함수이다. 함수 f_3 보다 더 많은 지역 최적해를 갖으며 그 분포가 원형으로 전역 최적해를 감싸는 듯한 모습을 하여 최적화 문제로 매우 어려운 문제이다.

본 논문에서 제안한 박테리아 협동 최적화 방법은 2절에서 설명한 것처럼 확률적으로 적합도가 높은 방향쪽을 향하나 무작위적인 방향설정이나 직선 이동 최대

스텝 등의 영향으로 무작위적인 이동 경향을 보이게 된다. 이런 것을 바이어스된 무작위 이동 (biased random walk) 이라고 하며 함수 f_1에서 좌표 -20, -20에 위치한 하나의 인공 대장균이 좌표 0, 0 최적 지점을 향해가는 것을 추적한 그림 2 (논문[11]에서 발췌)를 보면 보다 잘 이해할 수 있다.

모든 최적화 함수는 2차원 문제로서 각 차원별로 12비트에 해당하는 4,096 개의 영역으로 나눠 탐색영역을 구축하였다. 그러므로 모든 함수에서의 탐색공간은 $16,777,216 = (4096*4096)$이 된다. 박테리아 협동 최적화 알고리즘에 사용한 (B_n, B_m) 값과 (D_n, D_m) 값은 각각 $(3,9)$ 와 $(3,9)$이며 이는 논문[11]에서 얻은 의미 있는 값이다. 논문[12]에서 사용한 일정 퍼센트의 나쁜 개체를 무작위 적으로 생성한 개체로 대체하는 방법과의 성능 비교를 위하여 이 실험도 수행하였다. 이 실험에서는 논문[12]에서 대부분의 함수에서 대체로 좋은 성능을 보였던 50%의 나쁜 개체를 대체하는 것으로 설정하였다.

또한 단순 유전자 알고리즘과의 성능 비교를 위하여 똑 같은 문제를 유전자 알고리즘에 적용하여 실험을 해보았다. 가능한 한 개체 수, 최적화 탐색 공간 및 기타 파라미터를 맞추게끔 실험설계를 하였다. 유전자 알고리즘의 경우 대부분의 파라미터를 가장 무난하게 설정하는 값을 사용하였다. 즉 교배확률 $p_c = 0.5$, 돌연변이 확률 $p_m = 0.05$를 사용하였으며 룰렛 휠 선택방법을 사용하였다. 탐색공간을 같게 하기 위하여 x, y 두 축으로 각각 12비트로 코딩하였다. 개체 수도 인공 대장균의 개수와 동일하게 실험하였다.

실험은 초기 개체를 다르게 하여 10번씩 실험하여 평균을 구하였다. 표준편차도 구했지만 간략하게 나타내기 위하여 결과에는 넣지 않았다. 각 실험에서 개체가 최적해를 찾은 경우 이때의 세대 수를 기록하는 방식으로 실험하였다. 그러므로 세대 수가 작을수록 성능이 좋은 것이다. 실험결과는 표 2와 같다. 표 2에서 sGA[2]는 단순 유전자 알고리즘, sBCO[11]는 단순 박테리아 협동 최적화, sBCO_RR[12]는 낮은 등급 교체 박테리아 협

표 2. 실험 결과 (숫자는 10번 실험하여 최적해를 찾은 세대수를 평균한 것임)
Table 2. Experimental results (These are the averaged values of the numbers of generations finding optimal solutions after 10 runs).

함수	개체수	sGA	sBCO	sBCO_RR	sBCO_SC
f_1	100	10276.8	887.8	194.8	183.6
	200	9622.1	406.7	136.8	100.4
	300	4956.0	378.3	134.5	77.3
f_2	100	308776.0	976.9	260.0	238.7
	200	352490.8	777.1	187.3	135.2
	300	97377.0	644.6	148.8	95.5
f_3	100	14642.0	852.5	229.8	361.1
	200	13887.4	660.8	178.4	170.4
	300	15627.2	603.5	160.1	88.6
f_4	100	40676.5	838.9	737.1	696.8
	200	25782.0	689.3	413.4	283.8
	300	8992.5	517.7	334.2	326.0

$$f_1 = 3000 - 3(x^2+y^2), where\ -20 \le x,y \le 20$$

$$f_2 = e^{(-(\frac{x+3}{3})^2-(\frac{y+8}{3})^2)} + 0.8e^{(-(\frac{x-10}{4})^2-(\frac{y-6}{4})^2)} + 0.34e^{(-(\frac{x+10}{5})^2-(\frac{y-6}{4})^2)} + 0.19e^{(-(\frac{x+22}{4})^2-(\frac{y+25}{6})^2)} + 0.13e^{(-(\frac{x+1}{9})^2-(\frac{y-16}{6})^2)} + 0.10e^{(-(\frac{x+13}{8})^2-(\frac{y+6}{8})^2)} + 0.07e^{(-(\frac{x-11}{7})^2-(\frac{y+10}{8})^2)},\ where -20 \le x,y \le 20 \quad (1)$$

$$f_3 = \sum_{i=1}^{27} A_i e^{(-(\frac{x+B_i}{C_i})^2-(\frac{y+D_i}{E_i})^2)},\ where -20 \le x,y \le 20 \text{ and } A_i,\ B_i,\ C_i, \text{and} D_i \text{ are given as Table 1}$$

$$f_4 = 0.5 - \frac{\sin(\sqrt{x^2+y^2})\sin(\sqrt{x^2+y^2}) - 0.5}{(1.0+0.001(x^2+y^2))(1.0+0.001(x^2+y^2))},\ where -10 \le x,y \le 10$$

동 최적화 방법이며 SBCO_SC 는 본 논문에서 제안한 속도개념을 적용한 박테리아 협동 최적화 방법이다. 일단 모든 BCO 방법은 sGA 보다 성능이 좋았다. 또한 모든 경우에서 SBCO_RR 과 SBCO_SC 가 sBCO 보다 성능이 좋았다. 이는 sBCO 가 상대적으로 지역 최적해에 잘 빠지고 일단 빠지면 잘 나오지 못하는 것에 기인한 것으로 판단된다. sBCO_RR 과 sBCO_SC 를 비교해보면 많은 경우에 본 논문에서 제안한 sBCO_SC가 성능이 좋았다. 다만 함수 f_3 에서 개체수가 100일 때만이 sBCO_SC 가 sBCO_RR보다 성능이 떨어졌다. 논문[12]에서 보듯이 sBCO_RR 의 경우는 새로운 개체로 대체할 퍼센트를 적절히 선정해야한다. 그러나 본 논문에서는 그러한 과정이 없어도 더욱 좋은 성능을 보여주고 있다. 또한 속도를 조절하는 방법에 따라서 다양하며 더 성능이 좋은 방법을 개발할 수 있기 때문에 본 논문에서 제안한 방법이 더욱 효과적이라고 말할 수 있다.

IV. 결 론

본 논문에서는 박테리아 협동 최적화 알고리즘의 인공 대장균에 속도 개념을 적용하여 한 번에 이동하는 거리를 조절함으로서 성능을 향상시키는 방법을 제안하였다. 이를 위하여 인공 대장균들을 적합도에 따라서 등급을 매기고 등급이 낮은 개체는 전역 최적해와 멀리 있을 가능성이 높으므로 속력을 높게 하고 등급이 높은 개체는 전역 최적해 근방에 있을 가능성이 높으므로 속력을 낮게 하였다. 이러한 속력 개념은 적합도가 높은 개체는 근방을 세부적으로 탐색하게하고 적합도가 낮은 개체는 넓은 영역을 탐색하게 하여 지역 최적해에 빠지는 가능성을 낮추며 지역 최적해에 빠진 경우에 쉽게 빠져나오게 하여 성능을 향상시킨다. 4개의 함수 최적화 문제에 적용한 결과 예상했던 대로 기존의 단순 박테리아 협동 최적화는 물론 낮은 적합도를 갖는 인공 대장균의 일정 퍼센트를 무작위적인 새로운 인공 대장균으로 대체하는 방법보다도 더 성능이 좋음을 보았다.

결과적으로 본 논문에서 제안한 속력 개념을 적용한 박테리아 협동 최적화 방법이 효과적인 방법임을 확인하였다. 앞으로 추가적으로 대장균의 간접통신 방법인 정족수 센싱 (quorum sensing) 메커니즘을 추가적으로 적용하면 더욱 좋은 성능을 보일 것으로 기대된다.

참 고 문 헌

[1] D. Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison-Wesley, 1989.

[2] M. Srinivas and L. M. Patnaik, "Genetic Algorithms: A Survey," IEEE Computer Magazine, pp. 17-26, June 1994.

[3] R. C. Eberhart, Y. Shi, and J. Kennedy, "Swarm Intelligence," Morgan Kaufmann, 2001.

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, "Swarm Intelligence: From Natural to Artificial Systems," Oxford University Press, 1999.

[5] M. Dorigo and T. Stutzle, "Ant Colony Optimization," The MIT Press, 2004.

[6] M. Clerc, "Particle Swarm Optimization," ISTE Publishing Company, 2006.

[7] L. N. de Castro and J. Timmis, "Artificial Immune Systems: A New Computational Intelligence Approach," Oxford University Press, 2002.

[8] H. C. Berg and D. A. Brown, "Chemotaxis in escheichia coli analysed by three-dimensional tracking," Nature, vol. 239, pp. 500-504, 1972.

[9] M. Kim, S. Baek, S. H. Jung, and K.-H. Cho, "Dynamical characteristics of bacteria clustering by self-generated attractants," Computational Biology and Chemistry, vol. 31, pp. 328-334, Oct. 2007.

[10] T.-H. Kim, S. H. Jung, and K.-H. Cho, "Investigations into the design principles in the chemotactic behavior of Escherichia coli," BioSystems, vol. 91, pp. 171-182, Jan. 2008.

[11] S. H. Jung and T.-G. Kim, "A Novel Optimization Algorithm Inspired by Bacteria Behavior Patterns," Journal of Korean Institute of Intelligent Systems, vol. 18, pp. 392-400, June 2008.

[12] S. H. Jung, "Simple Bacteria Cooperative Optimization with Rank Replacement," Journal of Korean Institute of Intelligent Systems, vol. 19, pp. 432-436, June 2009.

저 자 소 개

정 성 훈(정회원)

1988년 한양대학교 전자공학과 (공학사)

1991년 KAIST 전기및전자공학과 (공학석사)

1995년 KAIST 전기및전자공학과 (공학박사)

1996년～현재 한성대학교 정보통신공학과

<주관심분야 : 소프트컴퓨팅, 시스템생물학, 뇌공학>