한국지능시스템학회 논문지 2009, Vol. 19, No. 3, pp. 432-436

Simple Bacteria Cooperative Optimization with Rank Replacement

Sung Hoon Jung

Department of Information and Communication Engineering, Hansung University

Abstract

We have developed a new optimization algorithm termed simple bacteria cooperative optimization (sBCO) based on bacteria behavior patterns [1]. In [1], we have introduced the algorithm with basic operations and showed its feasibility with some function optimization problems. Since the sBCO was the first version with only basic operations, its performance was not so good. In this paper, we adopt a new operation, rank replacement, to the sBCO for improving its performance and compare its results to those of the simple genetic algorithm (sGA) which has been well known and widely used as an optimization algorithm. It was found from the experiments with four function optimization problems that the sBCO with rank replacement was superior to the sGA. This shows that our algorithm can be a good optimization algorithm.

Key Words : Optimization, Bacteria Cooperative Optimization, Bacteria foraging, Bacteria chemotaxis

1. Introduction

Recently bio-inspired algorithms such as ant colony optimization, artificial immune system, and particle swarm optimization have been introduced and widely used for engineering applications [2-6]. These algorithms are based on the naturally optimized properties of natural organisms that have been evolved for few millenniums. In order to revise another bio-inspired algorithm, we focused on an Escherichia coli (often abbreviated to E. coli).

The E. coli, a kind of bacteria, usually live in a fluid (water) and swim by rotating several helical filaments called flagella. Their swimming consists of runs and tumbles [7,8]. If E. coli sense zero or negative gradients of attractant chemical molecules in regular temporal base on average, they tumble and new direction is randomly chosen. Otherwise, they run straight. From this biased random walk, E. coli can trace attractant chemical molecules for foraging. We have analyzed the foraging behaviors of E. coli and modeled their behavioral properties into behavior rules and decision rules in previous study [9]. Based on these rules, we have proposed a novel optimization algorithm termed simple bacteria cooperative optimization (sBCO) in [1].

In [1], our sBCO algorithm showed some possibility, but we could not be sure whether it could be a new good optimization algorithm because it had only basic operations and we didn't compare to the other, well known optimization algorithms. The sBCO consists of only three basic operations such as moving, sensing, decision with which artificial E. colis (AEs) find global optima. These three operations are not enough to show good performances because three operations can make AEs just iteratively search their neighbors step by step without additional operations for exploration. In sGA, the crossover operation can be viewed as an exploitation operation because it uses informations previously found and the mutation operation can be regarded as an exploration operation because it explores new search areas.

With this observation in mind, we adopt a new operation, rank replacement, in order to improve the performance of sBCO. In rank replacement, some percents of bad AEs are replaced with newly generated AEs. With rank replacement, sBCO can show some exploration effects similar to the mutation in sGA. This rank replacement can help the sBCO find a new area and also help the sBCO escape a local optimum area. Escaping a local optimum area is very important in most optimization algorithms because falling local optimum area makes optimization algorithms have very poor performances.

In order to measure the effects of rank replacement, we experimented our algorithm with four function optimization problems. Experimental results showed that the rank replacement could make our algorithm be a good optimization algorithm compared to the sGA. Our algorithm showed better performances about 40 times to about 2000 times than the sGA.

This paper is organized as follows. Section 2 describes the simple bacteria cooperative optimization with rank replacement. We deal with the experiments with four function optimization problems and discuss their results compared to those of sGA in section 3. We conclude this paper in section 4.

접수일자 : 2009년 4월 29일
완료일자 : 2009년 6월 5일
This research was financially supported by Hansung University in the year of 2008.

2. Simple Bacteria Cooperative Optimization with Rank Replacement

Our simple bacteria cooperative optimization (sBCO) algorithm is based on the behavioral patterns of E. colis. E. colis can do run or tumble according to the gradients of attractant chemical molecules. They decide their action every predefined movement using the density changes of attractant chemical molecules [9]. In order to make an optimization algorithm from these behavioral patterns, we have first made two kinds of rules, behavior rules and decision rules with artificial E. colis (AEs) on the discretized playground [1]. In the playground, AEs can move one unit to eight directions for a time step. We call the length of moving without turn of their direction the run count. Under this environment, the behavior rules are given:

(B1) AEs decide their actions for run or tumble every

Algorithm 1 Simple Bacteria Cooperative Optimization with Rank Replacement

// t : discrete time //
// r_c : the run count //
// B_n : the minimum number of runs to decide the actions of AE //
// B_m : the maximum number of runs to go straight without tumble //
// D_n, : the number of units for measuring current density of attractant molecules //
// D_m : the number of units for measuring previous density of attractant molecules //
// ρ_{D_n}, : the current density of attractant chemical molecules //
// ρ_{D_m} : the previous density of attractant chemical molecules //
// $E(t)$: artificial E. Colis (AEs) at time t //

```
1  t = 0
2  initialize E(t)
3       make AEs at random positions uniformly distributed within operating ranges
4       set initial directions of all AEs to random
5       set initial modes of all AEs to run
6       set r_c, ρ_Dn, and ρ_Dm of all AEs to zero
7       sense and store the amount of attractant chemical molecules at current position
8  while (not termination-condition)
9  do
10      t = t + 1
11      move E(t)
12           move each AE for one unit of playground to its direction
13           increase r_c of each AE
14      sense E(t)
15           sense and store the amount of attractant chemical molecules at current position
16           calculate ρ_Dn and ρ_Dm        ▷ decision rule (D1)
17      decide E(t)
18           if (r_c mod B_n) = 0  then  ▷ behavior rule (B1)
19                if ρ_Dn > ρ_Dm then      ▷ decision rule (D2)
20                     set mode to run
21                else
22                     set mode to tumble
23                end if
24           end if
25           if r_c = B_m          then          ▷ behavior rule (B2)
26                set mode to tumble
27           end if
28           if tumble mode then
29                set direction to random direction except for current direction and opposite direction
30                set run mode
31                set r_c = 0
32           end if
33      replace E(t)
34           sort E(t) with the amount of attractant chemical molecules on the current position
35           change ζ percents of bad AE to newly initialized AE
36 end
```

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

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

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

$$f_4 = 0.5 - \frac{\sin(\sqrt{x^2+y^2})\sin(\sqrt{x^2+y^2}) - 0.5}{(1.0+0.001(x^2+y^2))(1.0+0.001(x^2+y^2))}, \ where -20 \le x, y \le 20$$

B_n runs,

(B2) If their run counts become to $B_m (> B_n)$, then AEs must do tumble.

When AEs decide their actions, they compare the current density of attractant chemical molecules to previous density of those. If current density is larger than previous one, then they keep the run action until the run counts become to B_m. If the run count becomes to B_m, then they must do tumble. This second behavior rule somewhat helps sBCO not to fall local optimum. In order to decide their actions, AEs must know the current and previous densities of attractant chemical molecules. For this, decision rules are necessary. The decision rules are given:

(D1) AEs calculate the current density of attractant chemical molecules using the average values of those on D_n steps and the previous density using the average values of those on $D_m (> D_n)$ steps on the discretized playground.

(D2) If the current density of attractant chemical molecules is greater than the previous density, then the AE decides its action to run, otherwise, tumble.

Based on these behavior and decision rules, we devised a sBCO algorithm. You can find more detailed description about sBCO in [1]. The sBCO showed some possibility, but we could not be sure whether it could be a new good optimization algorithm because it had only simple basic operations and showed relatively poor performances.

As a new operation, we add rank replacement to the sBCO algorithm as shown in Algorithm 1. We call this algorithm simple bacteria optimization with rank replacement (sBCORR). Rank replacement is to replace bad AEs into newly generated AEs in order to search new areas and to escape local optima. This rank replacement increases the performances of sBCORR.

In the algorithm, the amount of attractant chemical molecules on the playground is given by an optimization function. Therefore, AEs move to optimum areas and eventually find the global optimum. In sBCORR, AEs are first initialized and then iteratively optimized by four main operations such as move $E(t)$, sense $E(t)$, decide $E(t)$, and replace $E(t)$ until one of AEs reaches to global optimum. If some AEs reach to the boundary of playground, then they randomly turn and go. It can also be viewed that the replace $E(t)$ operation is to mimic Darwinian natural selection.

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

3. Experimental Results

Our sBCORR was tested on four function optimization problems as shown in Equation 1.

Figure 1 shows the input and output relations of four functions. Function f_1 is a simple and unimodal function, which has its maximum at $x = y = 0$. Function f_2 is a relatively simple, but multimodal function, which has its maximum at $x = -3$, and $y = -8$. Function f_3 has many local optimum distributed broadly on the playground whose values are nearly the same as those of the global optimum. As a very difficult function for optimization, function f_4, sometimes called Mexican hat, has a lot of local optimum around the global optimum located at $x = y = 0$.

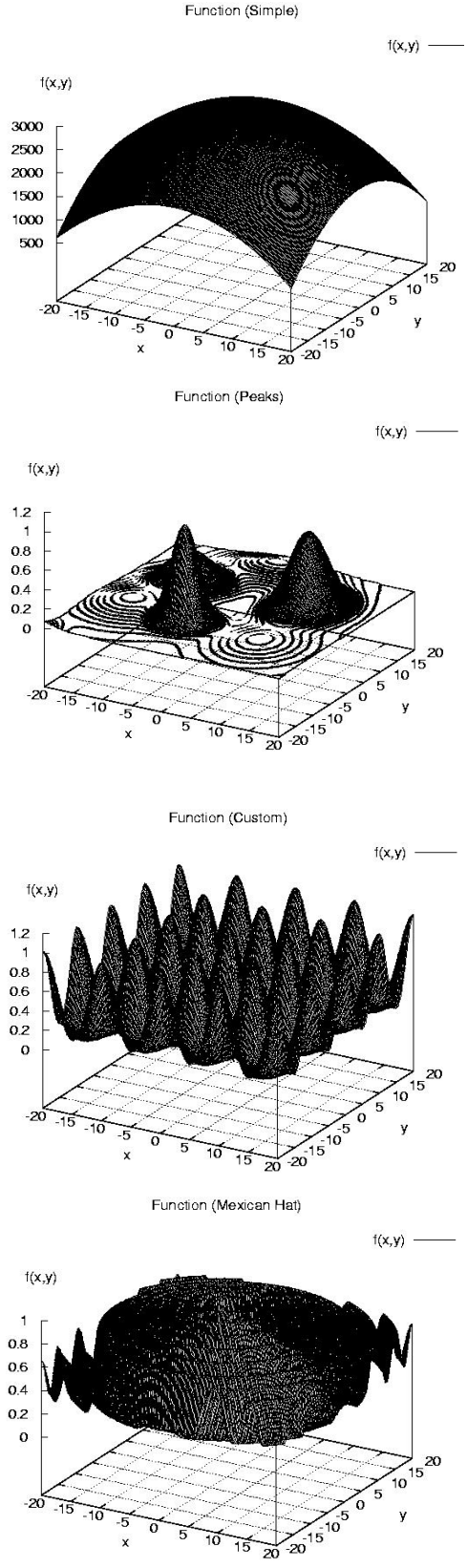

Figure 1. Experimental functions: f_1 (simple), f_2 (peaks), f_3 (custom), f_4 (Mexican hat)

The x and y axes of playground are discretized by 12 $(2^{12}=4096)$bits, respectively. Thus, the total search space is 16,777,216. We set the (B_n, B_m) and (D_n, D_m) to (3, 9) and (3, 9) respectively because it was revealed that the values were proper in our previous work [9]. We tested our algorithm with various ζ percents and various number of AEs. If one of AE reaches to global optimum, then its execution is stopped and the discrete time t is recorded.

In order to compare our sBCORR algorithm, a simple genetic algorithm (sGA) [10] was also tested under the same environments to our sBCORR algorithm. We set the crossover probability of sGA to 0.6 and the mutation probability of sGA to 0.05.

Table 2 shows the results of sGA and sBCORR. All results are average values of 10 runs with different random number seeds. In the results, we display only integer part of average values for simplicity and underline the best results of sBCORR. In order to show the effect of individuals, we used 10, 50, 100 number of AEs for sBCORR and 10, 50, 100 number of chromosomes for sGA. As shown in the results, our sBCORR algorithm is superior than the sGA about 40 times to about 2000 times.

We can also observe the other points from the results. First, sBCORR showed relatively stable performances according to the number AEs and replacement percents of bad AEs. This imply that sBCORR is a more reliable and robust method not to be perturbed its performances according to its parameters than the sGA. Second, about 50 percents of bad AEs showed good performances in most functions. This means that we don't need another method to select proper replacement percents.

4. Conclusion

In this paper, we proposed a bacteria cooperative optimization algorithm with rank replacement. From experiments with four function optimization problems it was found that our sBCO algorithm with rank replacement was superior than a simple genetic algorithm that has been well known and widely used to date. This implied that our sBCO algorithm with rank replacement could be a new good framework for optimization. Generally bacteria secrete quorum sensing molecules for detecting their quorum from the density of them and use it for communicating each other and detecting environmental changes. From this, they can more effectively do foraging.

In order to increase the performances of our algorithm, we will incorporate this quorum sensing mechanism into our algorithm as a further work. Also, we will introduce breeding of AEs like the sGA. This incorporation will make our algorithm be a truly cooperative algorithm.

Table 2. Experimental results

fn	no.	sGA	Replacement percents of bad AEs (ζ)								
			10	20	30	40	50	60	70	80	90
f_1	10	295595	582	506	506	376	335	335	427	298	476
	50	29553	387	190	220	286	205	194	222	195	245
	100	10276	272	242	207	186	179	180	157	147	151
f_2	10	97777	709	655	655	537	419	419	555	547	641
	50	174742	454	340	325	258	272	275	222	252	310
	100	308776	425	346	291	248	205	195	172	202	196
f_3	10	129226	1899	883	883	721	661	661	524	662	606
	50	25082	466	360	344	442	328	287	291	249	316
	100	14642	323	398	327	291	222	219	231	272	236
f_4	10	387265	29598	15167	15167	14829	8464	8464	10262	23170	47529
	50	53437	3902	2178	3652	4093	1769	3254	6320	5037	4341
	100	40676	1034	2150	1283	962	1364	1727	1734	2513	2667

References

[1] 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.

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

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

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

[5] M. Clerc, *Particle Swarm Optimization*. Publishing Company, 2006.

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

[7] 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.

[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] 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.

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

저 자 소 개

정성훈(Sung Hoon Jung)

1991년 : 한국과학기술원 전기및전자공학과 (공학석사)

1995년 : 한국과학기술원 전기및전자공학과 (공학박사)

1996년 : 한국과학기술원 전기및전자공학과 위촉연구원

1996년~현재 : 한성대학교 정보통신공학과 조교수,부교수,정교수

관심분야 : 진화연산, 신경망, 퍼지, 시스템생물학, 생물지능

E-mail : shjung@hansung.ac.kr