한국지능시스템학회 논문지 2008, Vol. 18, No. 3, pp. 392-400

A Novel Optimization Algorithm Inspired by Bacteria Behavior Patterns

Sung Hoon Jung* and Tae-Geon Kim*

* Department of Information and Communication Engineering, Hansung University

Abstract

This paper proposes a novel optimization algorithm inspired by bacteria behavior patterns for foraging. Most bacteria can trace attractant chemical molecules for foraging. This tracing capability of bacteria called chemotaxis might be optimized for foraging because it has been evolved for few millenniums. From this observation, we developed a new optimization algorithm based on the chemotaxis of bacteria in this paper. We first define behavior and decision rules based on the behavior patterns of bacteria and then devise an optimization algorithm with these behavior and decision rules. Generally bacteria have a quorum sensing mechanism that makes it possible to effectively forage, but we leave its implementation as a further work for simplicity. Thereby, we call our algorithm a simple bacteria cooperative optimization (BCO) algorithm. Our simple BCO is tested with four function optimization problems on various parameters of the algorithm. It was found from experiments that the simple BCO can be a good framework for optimization.

Key Words : Optimization, Bio-inspired engineering, Bacteria behavior patterns, Bacteria chemotaxis

1. Introduction

Bio-inspired engineering such as swarm intelligence and artificial immune system based on systems-level knowledges on an organism or cellular mechanisms has been recently focused for engineering applications [1-3]. One of the engineering application areas is optimization because a lot of engineering problems are related to optimizing parameters [4-7]. Bio-inspired optimization algorithms such as genetic algorithms, ant colony optimization (ACO), and particle swarm optimization [7-9] have been widely applied to scientific and engineering problems. Recently, two optimization algorithms based on the bacteria chemotaxis have been introduced [10-12].

In order to analyze the properties of bacteria, we also have investigated bacteria behavior patterns for foraging using Escherichia coli (often abbreviated to E. coli) which is one of the bacteria and made the model of behavior patterns in the literatures [13,14]. It was found from considerations that the model of behavior patterns of E. coli could be a good basis for a new optimization algorithm because their behavior patterns have shown good tracing capability for foraging. With this observation in mind, we devised a novel optimization algorithm based on the behavior patterns of E. coli.

Most bacteria, especially E. coli in this paper, can trace attractant chemical molecules for foraging. They usually live in a fluid (water) and swim by rotating several helical filaments called flagella which are controlled by membrane-embedded motors. Their swimming consists of runs and tumbles [15]. Run is the stages when an E. coli swims along a relatively straight line and tumble is when an E. coli's swimming direction changes. When an E. coli runs, all the flagella motors rotate in counterclockwise (CCW, viewed from outside of E. coli) and form a bundle. Otherwise, when an E. coli tumbles, a few motors change the rotation direction to clockwise (CW) and corresponding flagella leave the bundle, then the E. coli does erratic motion [16].

If an E. coli senses zero or negative gradients of attractant chemical molecules in regular temporal base on average, the E. coli tumbles and new direction is randomly chosen. This results in random walk. Otherwise, if an E. coli senses positive gradients of attractant chemical molecules, then the E. coli changes the rotation direction to CCW that makes the E. coli run to the direction of last tumble. Although the tumble direction is always random, the biased tumbling frequency allows the E. coli migrate toward favorable environment or avoid harmful chemicals. From this biased random walk, bacteria can trace attractant chemical molecules for foraging. This tracing capability of bacteria called chemotaxis might be optimized for foraging because it has been continuously evolved for few millenniums.

In order to devise an optimization algorithm from the behavior patterns of E. coli, we first make behavior rules and decision rules and then devise an optimization algorithm with those rules. Generally bacteria secrete quorum sensing molecules for detecting their quorum from the density of quorum sensing molecules and for communicating each other in order to deal with envi-

접수일자 : 2008년 2월 15일
완료일자 : 2008년 4월 18일
This research was financially supported by Hansung University in the year of 2008.

ronmental changes. This quorum sensing mechanism can be used as an indirect communication mechanism among bacteria as pheromone of ACO for cooperation. As a result, this quorum sensing mechanism helps bacteria effectively forage attractant chemical molecules. However, we didn't implement this quorum sensing mechanism for simplicity, so we call our optimization algorithm a simple bacteria cooperative optimization (BCO) algorithm. We will embed this mechanism into the simple BCO algorithm as a further work. We tested our simple BCO algorithm with four function optimization problems that have been widely used in optimization methods. We analyse and discuss experimental results.

This paper is organized as follows. Section 2 describes proposed simple BCO algorithm. In section 3, experimental results of BCO are discussed. This paper concludes in section 4.

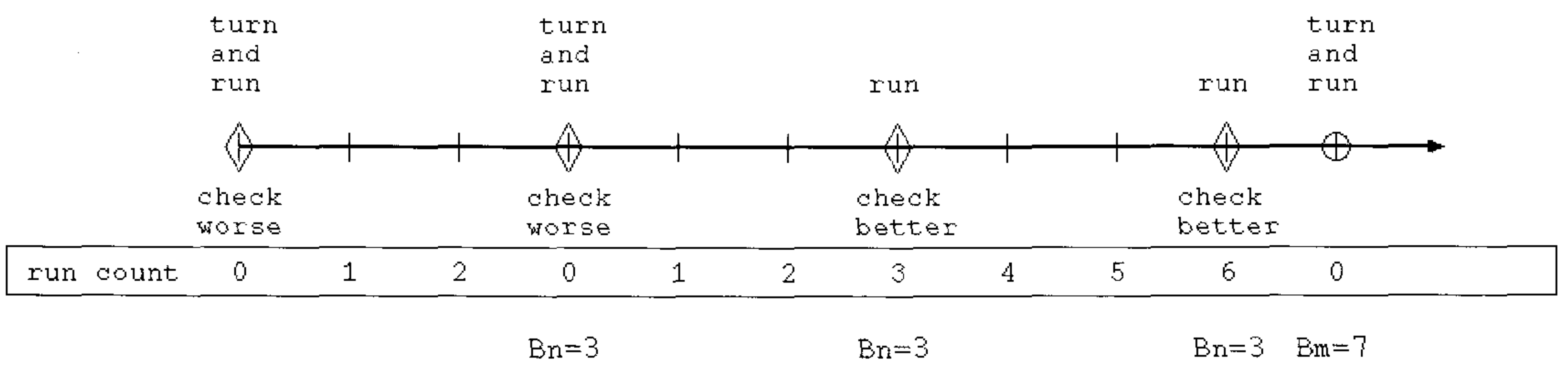
2. Simple Bacteria Cooperative Optimization

As described in previous section, an E. coli can do run or tumble according to the gradients of attractant chemical molecules, so we can consider the behavior patterns of E. coli into two aspects. First aspect is how often an E. coli decides its action and second aspect is how an E. coli senses the gradients of attractant chemical molecules and how an E. coli decides its action. From these two aspects, we derived two rules, behavior rules for the first aspect and decision rules for the second aspect.

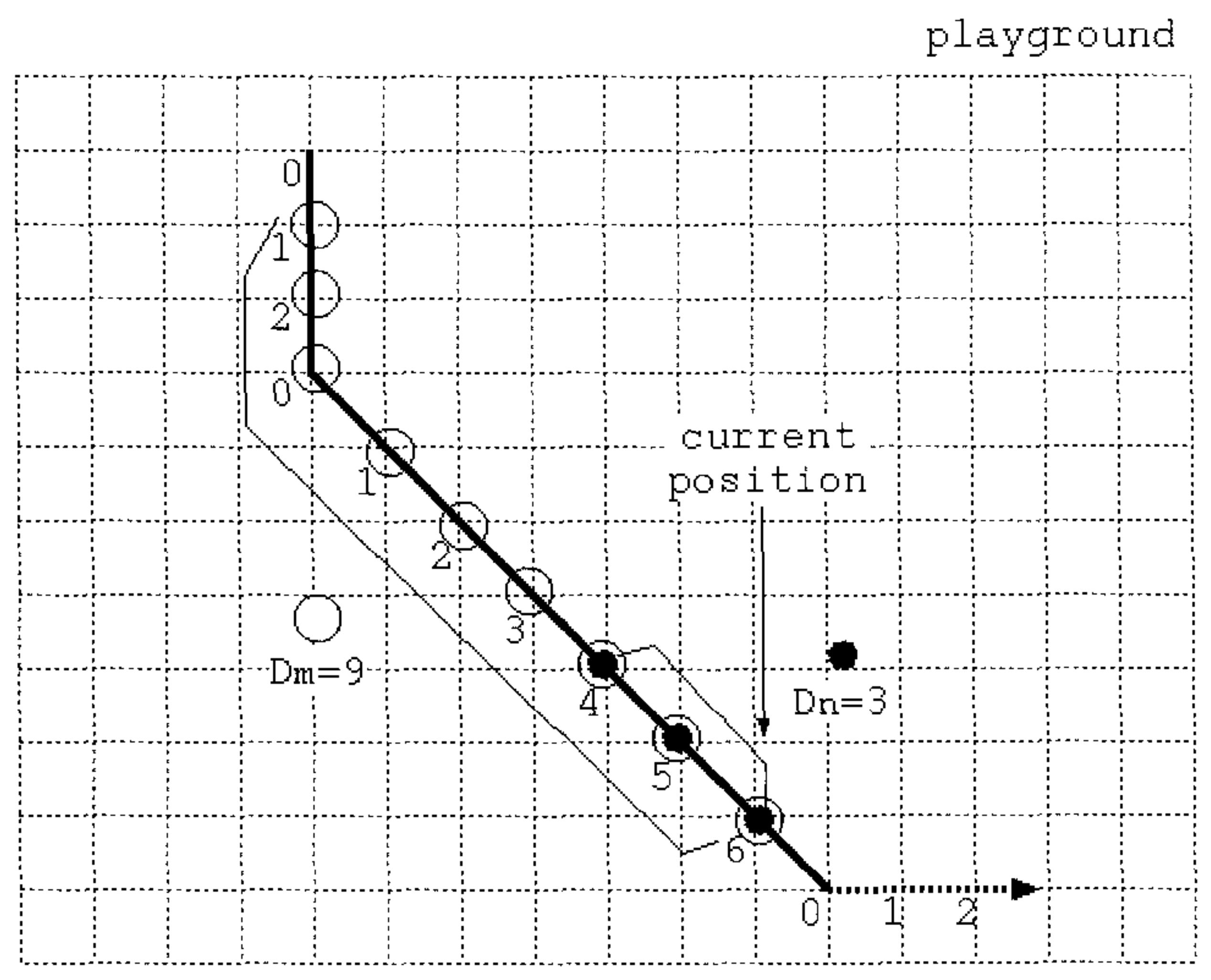
In order to model the behavior patterns of E. coli into behavior rules and decision rules, we assume that a modeled E. coli called artificial E. coli (AE) moves on a discretized area (called playground in this paper) at discrete time steps composed of minimum unit times. In the playground, AEs can move one unit to eight directions for a time step. We call the move without turn of its direction run and the length of runs the run count. Thus, if an AE changes its direction, then its run count becomes to zero. Under this discretized environment, behavior rules are given as follows.

(B1) The AE decides its action for run or tumble every B_n runs.

(B2) If the run count becomes to $B_m(> B_n)$, then the AE must do tumble.



(a)



(b)

Figure 1: A working example (a) behavior rules ($B_n = 3, B_m = 7$) (b) decision rules ($D_n = 3, D_m = 9$)

An AE decides its action at every B_n runs by the first behavior rule (B1). However, although an AE continuously decides run action, it can not go to same direction above maximum B_m runs by the second behavior rule (B2). This second behavior rule (B2) is very important to prevent the AE from falling local optimum that most bio-inspired optimization algorithms such as genetic algorithms and ant colony optimization have. If $B_m(>B_n)$ is too small, then it makes AEs take long time to reach global optimum. Otherwise, if B_m is too large, then it makes AEs fall in local optimum. Similarly, if B_n is too small, then an AE quickly follows local optimum and results in making it difficult to come out the local optimum. Otherwise, if B_n is too large, then it can not fine focus to global optimum. The two values of B_n and B_m should be carefully selected.

In order to decide its action, an AE should sense the gradients of attractant chemical molecules. Decision rules are related to how to sense the gradients of attractant chemical molecules and how to decide its action. Decision rules consist of two items as follows.

(D1) An AE sets current (previous) density of attractant chemical molecules to the average value of recent D_n $(D_m(>D_n))$ number of sensed attractant chemical molecules 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.

We depict a working example of behavior and decision rules in case of $B_n=3$, $B_m=7$ and $D_n=3$, $D_m=9$ in Figure 1. As shown in Figure 1 (a),

```
Algorithm 1 Simple Bacteria Cooperative Optimization (BCO)
// $t$ : discrete time //
// $r_c$ : the run count //
// $B_n$, $B_m$ : the minimum and maximum runs //
// $D_n$, $D_m$ : the number of units for measuring current and previous densities of attractant chemical molecules
// $\rho_{D_n}$, $\rho_{D_m}$ : the current and previous densities of attractant chemical molecules //
// $E(t)$ : artificial E. Colis (AEs) at time $t$ //
$t=0$
initialize $E(t)$
    make AEs at random positions uniformly distributed within operating ranges
    set initial directions of all AEs to random
    set initial modes of all AEs to run
    set $r_c$, $\rho_{D_n}$, $\rho_{D_m}$ and of all AEs to zero
    sense and store the amount of attractant chemical molecules at current position
while (not termination-condition)
do
    $t=t+1$
    move $E(t)$
        move each AE for one unit of playground to its direction
        increase $r_c$ of each AE
    sense $E(t)$
        sense and store the amount of attractant chemical molecules at current position
        calculate $\rho_{D_n}$ and $\rho_{D_m}$        ▷ decision rule (D1)
    decide $E(t)$
        if $(r_c \bmod B_n)=0$ then   ▷ behavior rule (B1)
            if $\rho_{D_n} > \rho_{D_m}$ then    ▷ decision rule (D2)
                set mode to run
            else
                set mode to tumble
            end if
        end if
        if $r_c$ = $B_m$     then        ▷ behavior rule (B2)
            set mode to tumble
        end if
        if tumble mode then
            set direction to random direction except for current direction and opposite direction
            set run mode
            set $r_c=0$
        end if
end
```

an AE decides its action at every 3 runs and it unconditionally tumbles if the run count becomes to 7 without considering the gradient of attractant chemical molecules. Figure 1 (b) explains how to calculate the current and previous density of attractant chemical molecules. Let us assume that an AE is on the current position of Figure 1 (b), then the AE should decide its action because of $B_n = 3$. In order to decide its action, the AE should calculate the current and previous densities. The current (previous) density is calculated to the average value of attractant chemical molecules on the black circles (white circles) because of $D_n = 3$ ($D_m = 9$). In the example, the AE decides its action to run because the current density of attractant chemical molecules is greater than the previous one.

Table 1: Parameters of function f_4

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

Based on these behavior and decision rules, we devise a simple BCO algorithm as shown in Algorithm 1. In simple BCO, AEs are first initialized and iteratively optimized by three main operations such as move $E(t)$, sense $E(t)$, and decide $E(t)$. In initialize $E(t)$, AEs are generated at random positions uniformly distributed within operating ranges in a playground. Their directions are randomly set and their mode is set to run and all parameters of them are set to zero. AEs sense and store the amount of attractant chemical molecules at current position. The amount of attractant chemical molecules on the playground is given by an optimization function. AEs in a playground move, sense, and decide its direction iteratively until some of them reach to global optimum. If an AE reaches the boundary of playground, then it randomly turns and goes.

3. Experimental Results

Our simple BCO was tested on typical four function optimization problems that have been used at previous papers [17-19]. The four functions are given in Equation 1. Figure 2 shows the input and output relations of four functions. Function f_1 is a very simple and unimodal function, which has its maximum at $x = y = 0$. Function f_2 is a relatively simple and multimodal function, which has its maximum at $x = -3$, and $y = -8$. As a very difficult function for optimization, function f_3, sometimes called Mexican hat function, has a lot of local optimum around the global optimum located at $x = y = 0$. Since it has a lot of local optimum around the global optimum, optimization algorithms can easily fall into the local optimum and rarely come out the local optimum. As another difficult type of optimization, function f_4 has many local optimum whose values are nearly the same as those of the global optimum. Function f_4 is more difficult to come out the local optimum than the function f_3 because the shape of local optimum of f_4 is peak.

In order to apply our simple BCO algorithm to function optimization, the x, y two dimensional playground is first discretized and then attractant chemical molecules are given corresponding to the optimizing function on the discretized playground. After that, the predefined number of AEs are generated on the discretized playground at random positions uniformly distributed within operating ranges. The generated AEs swim one unit at a time step according to their current directions. We call it one iteration that all AEs move one unit. If the number of runs of an AE becomes to B_n, then it decides its action for run and tumble. If it decides its action to run, then it continuously goes to the current direction. Otherwise, if tumble, then it changes its direction to the other directions except for current direc-

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

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

$$f_3 = 0.5 - \frac{\sin(\sqrt{x^2+y^2})\sin(\sqrt{x^2+y^2}) - 0.5}{(1.0+0.001(x^2+y^2))(1.0+0.001(x^2+y^2))},\ where -20 \le x,\ y \le 20$$

$$f_4 = \sum_{i=1}^{27} A_i e^{(-(\frac{x+B_i}{C_i})^2 - (\frac{y+D_i}{E_i})^2)},\ where -20 \le x,\ y \le 20 \text{ and } A_i,\ B_i,\ C_i, \text{ and } D_i \text{ are given as Table 1}$$

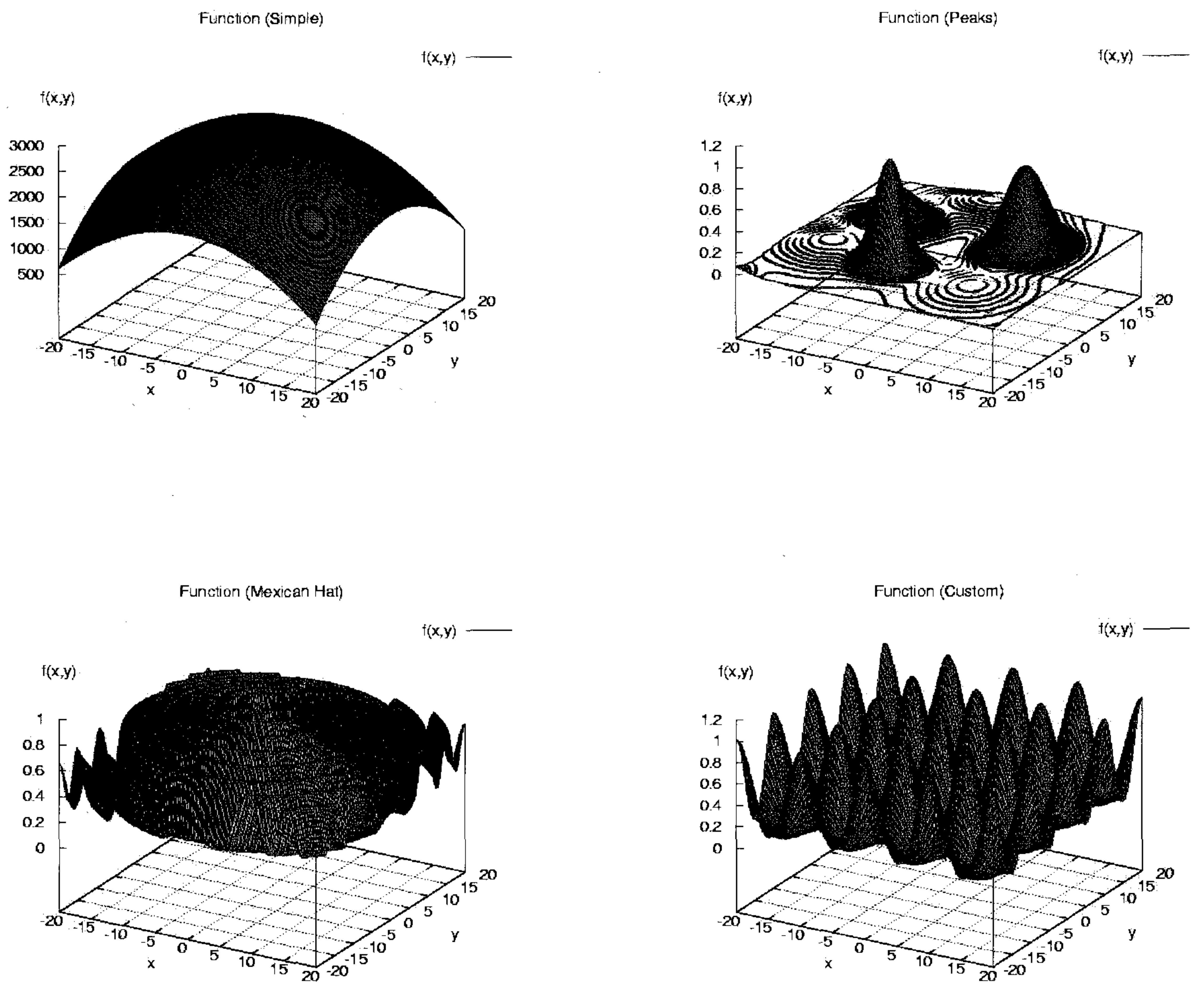


Figure 2: Experimental functions: f_1 (simple), f_2 (peaks), f_3 (Mexican hat), f_4 (custom)

tion and the opposite direction of current direction. An AE decides its action to run if the average value of chemical molecules on the D_n number of recently visited units is greater than that on the $D_m (> D_n)$ number of recently visited units. Otherwise, then it decides its action to tumble. From this, AEs tend to go to the direction that attractant chemical molecules increase. If the run count of an AE becomes to $B_m (> B_n)$, then it should do tumble even if it goes to good direction in order for preventing from falling local optimum. This results in biased random walk. Figure 3 shows this biased random walk in the simple function f_1. An AE that starts from the $(-20, -20)$ position goes to the center $(0, 0)$ position. The x and y axis of playground is discretized by $8 bits (= 256)$, respectively. Thus, the total search space is 65536. In order to measure the performances according to the parameters of behavior and decision rules, we measured the performances on $B_n = 1, 2, 3, 4, 5$, $B_m = 7, 8, 9, 10, 11$, $D_n = 1, 2, 3, 4, 5$, $D_m = 7, 8, 9, 10, 11$, respectively. Since the number of AEs is also an important parameter for performances, we experimented with the 10, 100,

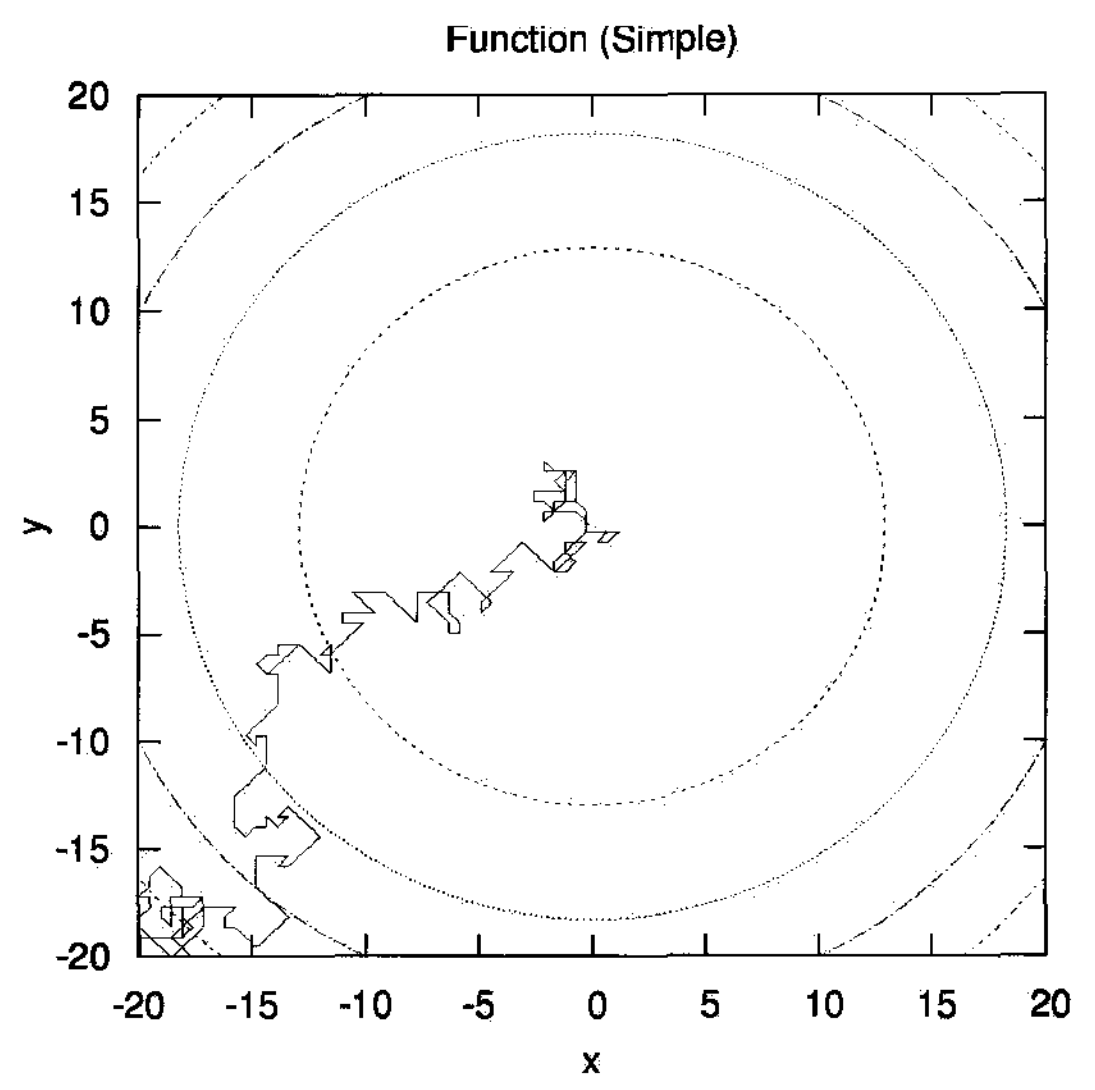
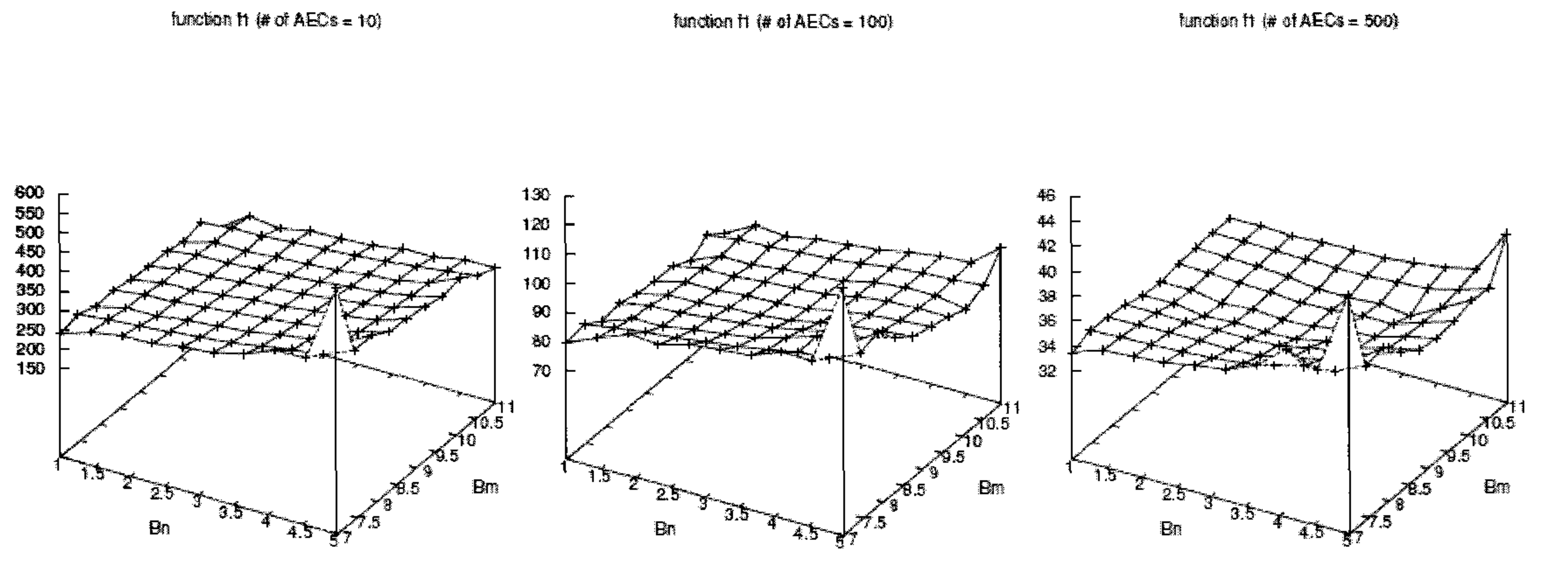
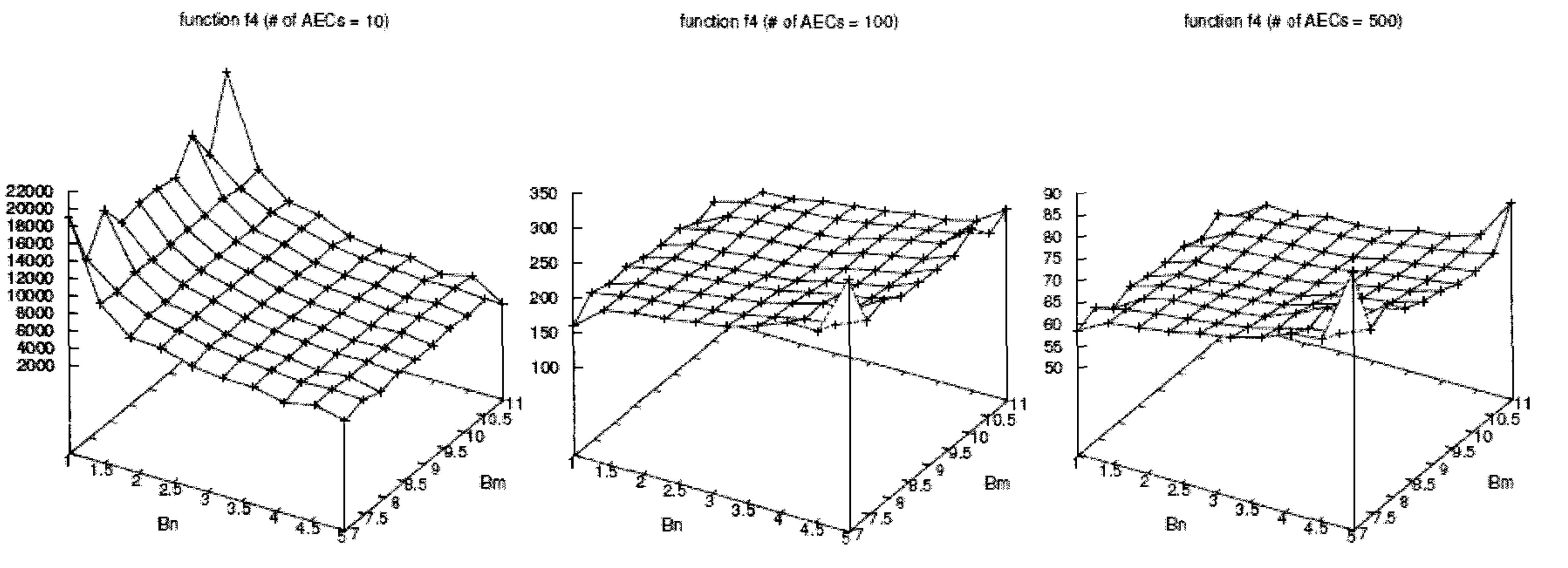


Figure 3: Trajectory of an AE for function 1



(a)



(b)

Figure 4: Performance graphs according to B_n and B_m (a) f_1 (b) f_4

500 number of AEs, respectively. For each experiment, the iteration number that the simple BCO finds global optimum is recorded and this number is used for the measure of performances.

Table 2 shows experimental results according to the number of AEs. For one parameter set of B_n, B_m, D_n, and D_m, we measured the results of ten number of experiments with different random number seeds. Therefore, the total number of experiments are $6250(=5(B_n)\times 5(B_m)\times 5(D_n)\times 5(D_m)\times 10)$. If the simple BCO falls into local optimum, then it is difficult to come out the local optimum especially in case of complex and multimodal functions such as f_3 and f_4. In this case, the simple BCO may not finish its work to find global optimum. For this case, we set the maximum iteration number to 400,000. We regard it fail when the simple BCO can not find the global optimum within the maximum iteration number. As shown in Table 2, the simple BCO has no fail in case of f_1 because it is an unimodal function (in other words, it has no local optimum). Even if function f_2 has six number of local peaks except for one global peak, it is relatively easy that the simple BCO comes out the local peaks. Thus, the function f_2 has also no fail. On the other hand, function f_3 and f_4 have 4, 110 fails when the number of AEs is 10, respectively. This is because function f_3 has local optimum around global optimum and function f_4 has many local peaks near global optimum. Since the shape of local optimum of function f_3 is not peaks, it is relatively easy for the simple BCO to come out the local optimum. On the other hand, the shape of local optimum of function f_4 is peaks, escaping the local optimum is more difficult than the case of the function f_3. Even if an optimization function has many local optimum, the simple BCO finishes its work within a maximum iteration in the most cases if the number of AEs is large. This is because the probability that initial AEs are generated around the global optimum is large

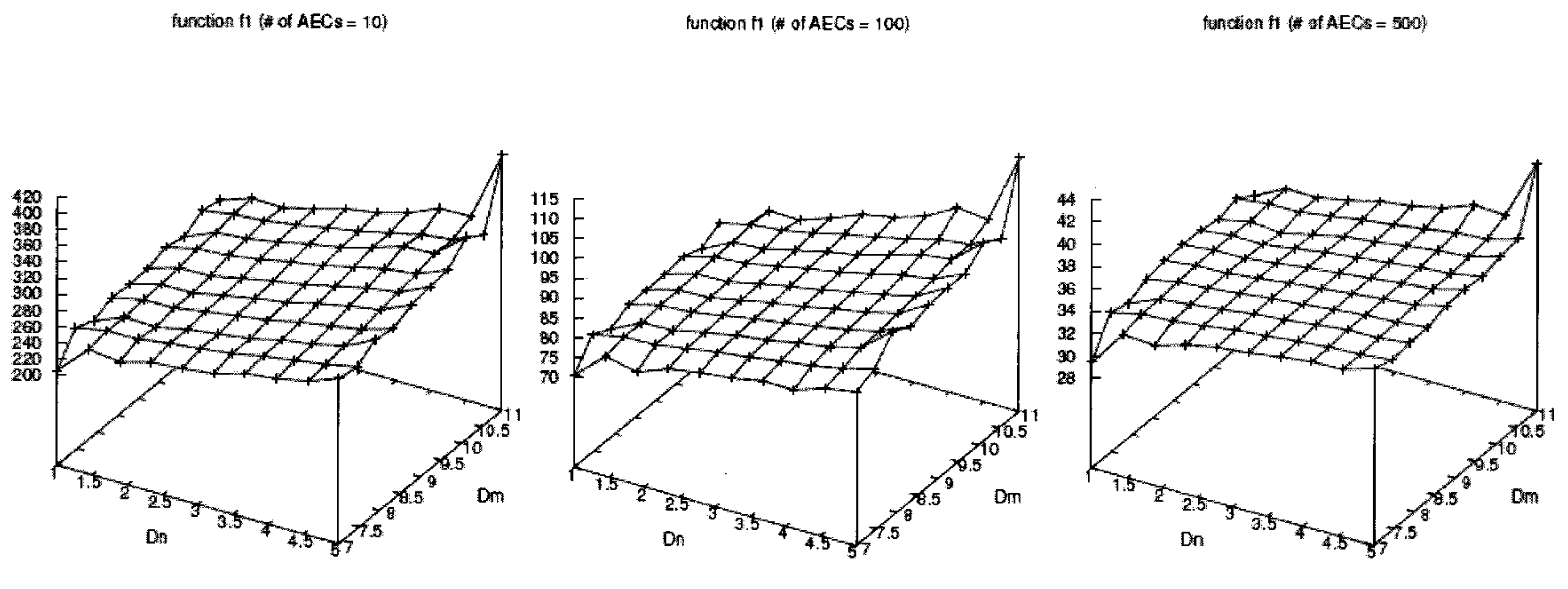
Table 2: Experimental results

function	# of AEs	average	standard deviation	success	fail
f_1	10	274.10	186.70	6250	0
	100	84.16	55.99	6250	0
	500	34.45	21.72	6250	0
f_2	10	5421.77	9082.36	6250	0
	100	301.10	334.24	6250	0
	500	56.68	55.31	6250	0
f_3	10	2404.37	12521.40	6246	4
	100	164.35	142.87	6250	0
	500	49.35	40.58	6250	0
f_4	10	7311.44	25479.05	6140	110
	100	208.69	227.70	6250	0
	500	63.79	41.98	6250	0

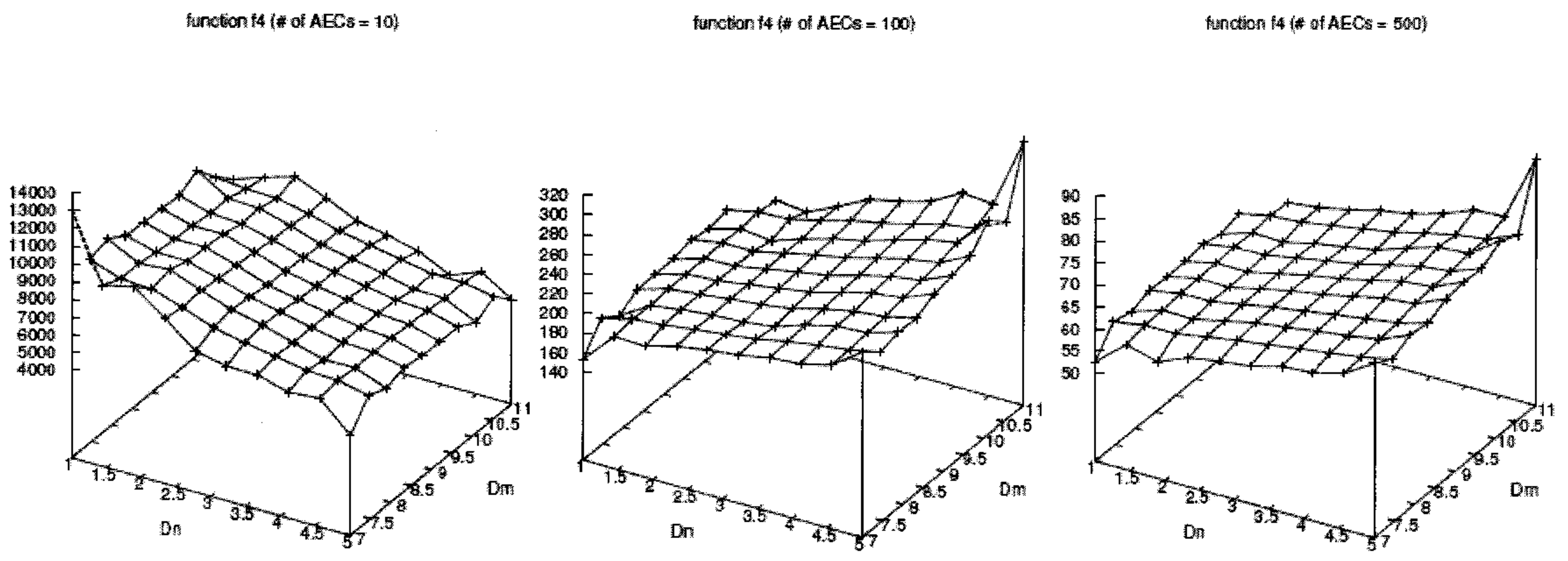
and some AEs can have a chance to come out the local optimum.

The standard deviations of all results except for two cases, 10 AEs of f_3 and 10 AEs of f_4, are very close to the average values. This indicates that the performances of simple BCO are not quite dependent on the parameters of behavior and decision rules and the random number seeds. In the two cases, simple BCO sometimes takes long time to find global optimum because it has no enough AEs to come out local optimum. If initial AEs are located to near global optimum, then the simple BCO can easily find the global optimum even if it has small number of AEs. Otherwise, the simple BCO has very difficulty to find the global optimum because it is not easy to get out of local optimum especially when the number of AEs is small.

In order to observe the effects of B_n, B_m, D_n, and D_m, we depict the performance graphs of function f_1 for simple function and function f_4 for complex function



(a)



(b)

Figure 5: Performance graphs according to D_n and D_m (a) f_1 (b) f_4

as shown in Figure 4 and 5, respectively. As shown in Figure 4, the simple BCO shows relatively similar performances on the simple function f_1, but it shows considerably large differences of performances on the complex function f_4. So we should select the values of B_n and B_m for complex functions more carefully than those for simple functions. The values of B_n and B_m in simple functions can not largely affect to the performances because AEs can easily approach to the global optimum. However, bad values of B_n and B_m in complex functions make it difficult for the simple BCO to come out local optimum especially when the number of AEs is small. The performances of simple function f_1 are good at $B_n = 1$ and $B_m = 7$ and bad at $B_n = 5$ and $B_m = 7$ in the three experiments. This is because fast decision for climbing hill is best for the simple function. However, fast decision in the complex function f_4 especially in the case of small AEs enables the simple BCO to fall into local optimum and not to come out the local optimum. If the number of AEs is enough to cover the playground, then fast decision does not make the performances of simple BCO degrade because some AEs may be generated near global optimum. Figure 5 shows the performances of simple BCO according to the parameters of decision rules, D_n and D_m. Overall performances are similar to those of the cases of B_n and B_m, but the performances are bad at $D_n = 5$ and $D_m = 11$ in this result. This may be because the larger D_m is, the more the simple BCO confuses to decide its action. As a result, we should carefully select the parameters of B_n, B_m, D_n, and D_m to fast go to global optimum and to easily come out local optimum.

4. Conclusion

In this paper, we proposed a novel optimization algorithm inspired by E. coli's behavior patterns. The behaviors patterns of E. coli for foraging naturally optimized for few millenniums were modeled to behavior rules and decision rules. Based on these behavior and decision rules, we devised a simple bacteria cooperative optimization (BCO) algorithm on the discretized playground. We tested our algorithm with four function optimization problems under various parameters of our algorithm. It was found from experiments that the simple BCO could be a good framework for optimization. In this simple BCO, we did not employ the quorum sensing mechanism of bacteria to our algorithm, but we will embed the mechanism into our algorithm as a further work. Additionally, the other various methods to increase the performances of the simple BCO algorithm such as generating offsprings and adopting variable run steps will be introduced. Also, the performances of BCO will be compared to those of the other existing bio-inspired optimization algorithms such as genetic algorithms, ant colony optimization, and particle swarm optimization.

참 고 문 헌

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

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

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

[4] D. B. Fogel, "An Introduction to Simulated Evolutionary Optimization," *IEEE Transactions on Neural Networks*, vol. 5, pp. 3-14, Jan. 1994.

[5] W.-S. Jwo, C.-W. Liu, and C.-C. Liu, "'Large-scale optimal VAR planning by hybrid simulated annealing/genetic algorithm,"' *International Journal of Electrical Power and Energy Systems*, vol. 21, pp. 39-44, Jan. 1999.

[6] C. Xudong, Q. Jingen, N. Guangzheng, Y. Shiyou, and Z. Mingliu, "An Improved Genetic Algorithm for Global Optimization of Electromagnetic Problems,'' *IEEE Transactions on Magnetics*, vol. 37, pp. 3579-3583, Sept. 2001.

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

[8] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

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

[10] Y. Liu and K. M. Passino, "Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors," *Journal of Optimization Theory and Applications*, vol. 115, pp. 603-628, Dec. 2002.

[11] K. M. Passino, "Biomimicry of Bacterial Foraging for Distributed Optimization and Control," *IEEE Control Systems Magazine*, vol. 22, pp. 52-67, June 2002.

[12] S. D. Muller, J. Marchetto, S. Airaghi, and P. Koumoutsakos, "Optimization Based on Bacterial Chemotaxis," *IEEE Transactions on Evolutionary Computation*, vol. 6, pp. 16-29, Feb. 2002.

[13] 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.

[14] 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.

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

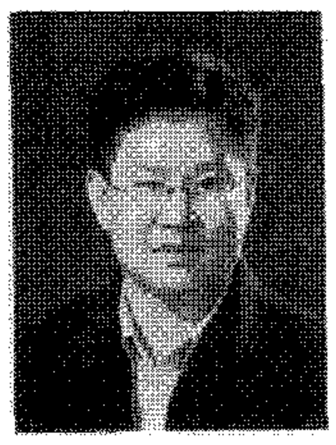
[16] L. Turner, W. S. Ryu, and H. C. Berg, "Real-time imaging of fluorescent flagellar filaments," *Journal of Bacteriology,* vol. 182, pp. 2793-2801, May 2000.

[17] K. DeJong, *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, 1975.

[18] J. Andre, P. Siarry, and T. Dognon, "An improvement of the standard genetic algorithm fighting premature convergence in continuous optimization," *Advances in engineering software,* vol. 32, no. 1, pp. 49-60, 2001.

[19] S. H. Jung, "Queen-bee evolution for genetic algorithms," *Electronics Letters,* vol. 39, pp. 575-576, Mar. 2003.

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

김태건(Tae-Geon Kim)

2006년 : 한성대학교 정보통신공학과 (공학사)

2006년~현재 : 한성대학교 일반대학원 정보통신공학과 석사과정

관심분야 : 진화연산, 신경망, 퍼지, 생물지능

E-mail : like.sunshine@gmail.com