Improved Bacterial Foraging Optimization with Social Cooperation and Adaptive Step Size

Xiaohui Yan[1,2,*], Yunlong Zhu[1], Hanning Chen[1], and Hao Zhang[1,2]

[1] Key Laboratory of Industrial Informatics, Shenyang Institute of Automation, Chinese Academy of Sciences, 110016, Shenyang, China
[2] Graduate School of the Chinese Academy of Sciences, 100039, Beijing, China
{yanxiaohui,ylzhu,chenhanning,zhanghao}@sia.cn

Abstract. This paper proposed an Improved Bacterial Foraging Optimization (IBFO) algorithm to enhance the optimization ability of original Bacterial Foraging Optimization. In the new algorithm, Social cooperation is introduced to guide the bacteria tumbling towards better directions. Meanwhile, adaptive step size is employed in chemotaxis process. The new algorithm is tested on a set of benchmark functions. Canonical BFO, Particle Swarm Optimization and Genetic Algorithm are employed for comparison. Experiment results show that the IBFO algorithm offers significant improvements over the original BFO algorithm and is a competitive optimizer for numerical optimization.

Keywords: bacterial foraging optimization, social cooperation, adaptive search strategies.

1 Introduction

Bacterial Foraging Optimization (BFO) is a novel swarm intelligence algorithm first proposed by Passino in 2002 [1]. It is inspired by the foraging and chemotactic behaviors of bacteria. Recently, BFO algorithm and its variants have been used for many numerical optimization [2] or engineering optimization problems [3-4].

However, original BFO algorithm has some weaknesses. First, the tumble angles are generated randomly. Useful information can't be shared between bacteria. Second, the step size in the original BFO is a constant. If the step size is large at the end stage, it is hard to converge to the optimal point. In this paper, we proposed an Improved Bacterial Foraging Optimization (IBFO). Two adaptive strategies are used in IBFO to improve its optimization ability. First, social cooperation is introduced to enhance the information sharing between bacteria. Then, adaptive step size is employed, which could make the bacteria use different search step sizes in different stages.

The rest of the paper is organized as followed. Section 2 introduces the original BFO algorithm. In section 3, the proposed IBFO algorithm is described in detail. In Section 4, the IBFO algorithm is tested on a set of benchmark functions compared with several other algorithms. Results are presented and discussed. Finally, conclusions are drawn in Section 5.

* This research is partially supported by National Natural Science Foundation of China 61174164, supported by National Natural Science Foundation of China 61003208 and supported by National Natural Science Foundation of China 61105067.

D.-S. Huang et al. (Eds.): ICIC 2012, LNCS 7389, pp. 634–640, 2012.

2 Original Bacterial Foraging Optimization

The *E. coli* bacterium is one of the earliest bacterium which has been researched. It has a plasma membrane, cell wall, and several flagella which are randomly distributed around its cell wall 1. By the rotation of the flagella, *E. coli* can "tumble" or "run" in the nutrient solution. By simulating the foraging process of bacteria, Passino proposed the BFO algorithm. The main steps of BFO are explained as followed.

2.1 Chemotaxis

In BFO, the position updating which simulates the chemotaxis procedure is used the Eq. (1) as followed. θ^t is the position of the bacterium in the *t*th chemotaxis step. *C(i)* presents the step size. $\Phi(i)$ is a randomly produced unit vector which stands for the tumble angle.

$$\theta_i^{t+1} = \theta_i^t + C(i)\phi(i) \tag{1}$$

In each chemotactic step, the bacterium generated a tumble direction firstly. Then the bacterium moves in the direction using Eq. (1). If the nutrient concentration in the new position is higher than the last position, it will run one more step in the same direction. This procedure continues until the nutrient get worse or the maximum run step is reached. The maximum run step is controlled by a parameter called N_s.

2.2 Reproduction

For every N_c times of chemotactic steps, a reproduction step is taken in the bacteria population. The bacteria are sorted in descending order by their nutrient values. Bacteria in the first half of the population will split into two. Bacteria in the residual half of the population die. By this operator, individuals with higher nutrient are survived and reproduced, which guarantees the potential optimal areas are searched more carefully.

2.3 Eliminate and Dispersal

After every N_{re} times of reproduction steps, an eliminate-dispersal event happens. For each bacterium, it will be moved to a random place according to a certain probability, known as P_e. This operator enhances the diversity of the algorithm.

3 Improved Bacterial Foraging Optimization

3.1 Social Cooperation

As we known, swarm intelligence is emerged by the cooperation of simple individuals [5]. However, the social cooperation hasn't been used in original BFO algorithm. In chemotactic steps, the tumble directions are generated randomly. Information carried by the bacteria in rich-nutrient positions is not utilized. In our IBFO, the tumble directions are generated using Eq. (2). Where θ_{gbest} is the global best of the population found so far. $\theta_{i,\ pbest}$ is the *i*th bacterium's personal historical best. The tumble direction is then normalized as unit vector and the position updating is still using the Eq. (1).

$$\Delta_i = (\theta_{gbest} - \theta_i) + (\theta_{i,pbest} - \theta_i) \tag{2}$$

The direction generating equation is similar with the velocity updating equation of PSO algorithm [6]. They all used the global best and personal best. However, they are not the same. First, there is no inertia term in Eq. (2). In chemotactic steps of bacteria, inertia term will enlarge the difference of between θ_{gbest}, $\theta_{i,\ pbest}$ and the current position tremendously. Second, there are no learning factors in Eq. (2) as the direction will be normalized to unit vector. By social cooperation, the bacteria will move to better areas with higher probability as good information is fully utilized.

3.2 Adaptive Step Size

As it mentioned above, the constant step size will make the population hard to converge to the optimal point. In an intelligence optimization algorithm, it is important to balance its exploration ability and exploitation ability. Generally, in the early stage of an algorithm, we should enhance the exploration ability to search all the areas. In the later stage of the algorithm, we should enhance the exploitation ability to search the good areas intensively.

There are various step size varying strategies [7-8]. In IBFO, we use the decreasing step size. The step size will decrease with the iteration, as shown in Eq. (3). C_s is the initial step size. C_e is the ending step size. *NowEva* is the current function evaluations count. *TotEva* is the total function evaluations. In the early stage of IBFO algorithm, we use larger step size to guarantee the exploration ability. And at the end stage, smaller step size is used to make sure the algorithm can converge to the optimal point.

$$C = C_s - (C_s - C_e) \times NowEva / TotEva \tag{3}$$

4 Experiments

In this section, we tested the optimization ability of IBFO algorithm on six benchmark functions. Original BFO, PSO and Genetic Algorithm (GA) were employed for comparison.

4.1 Benchmark Functions

The six benchmark functions are listed in Table 1. They are widely adopted by other researchers to test their algorithms in many works [9-10]. Dimension of all functions are 20.

To compare algorithms fairly, we use number of function evaluations (FEs) as a measure criterion in this paper. It is also used in many other works [11-12]. All algorithms were terminated after 60,000 function evaluations.

4.2 Parameter Settings for the Involved Algorithms

The population sizes S of all algorithms are 50. In original BFO and IBFO algorithm, the parameters are set as followed: N_c=50, N_s=4, N_{re}=4, N_{ed}=10, P_e=0.25, C=0.1, S_r=S/2=25. The initial step size of IBFO C_s=0.1(Ub-Lb), ending step C_e=0.00001(Ub-Lb) where Lb and Ub refer the lower bound and upper bound of the variables of the problems. In PSO algorithm, ω decreased from 0.9 to 0.7. $C1$=$C2$=2.0 [13]. V_{min}=0.1×Lb, V_{max}=0.1×Ub. In GA, P_c is 0.95 and P_m is 0.1.

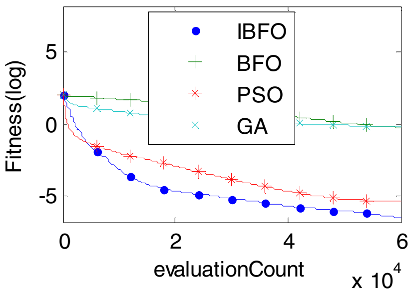
Table 1. Benchmark functions used in the experiment

Function	Formulation	Variable ranges	f(x*)
Sphere	$f_1(x)=\sum_{i=1}^{D} x_i^2$	[-5.12, 5.12]	0
Rosenbrock	$f_2(x)=\sum_{i=1}^{D-1}\left(100(x_i^2-x_{i+1})^2+(1-x_i)^2\right)$	[−15, 15]	0
Rastrigin	$f_3(x)=\sum_{i=1}^{D}\left(x_i^2-10\cos(2\pi x_i)+10\right)$	[-10, 10]	0
Ackley	$f_4(x)=20+e-20e^{\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)}-e^{\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)}$	[−32.768, 32.768]	0
Griewank	$f_5(x)=\frac{1}{4000}\left(\sum_{i=1}^{D}x_i^2\right)-\left(\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})\right)+1$	[−600, 600]	0
Schwefel2.22	$f_6(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	[-10,10]	0

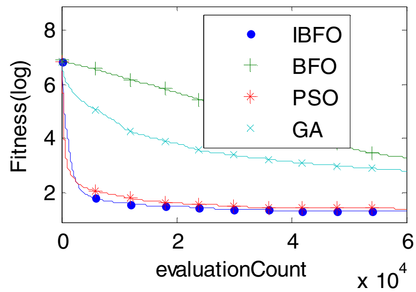
4.3 Experiment Results and Statistical Analysis

The results of IBFO, BFO PSO and GA on the benchmark functions are listed in Table 2. Best values of them on each function are marked as bold. Convergence plots of the algorithms on these functions are shown in Fig. 1.

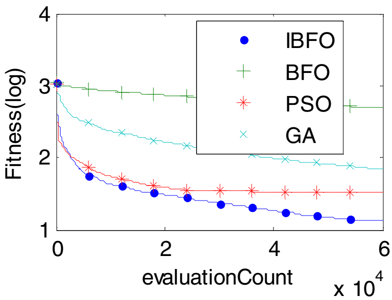
It is clear from table 2 that IBFO obtained the best values on five of all six functions. PSO is best on the rest one. BFO and GA performed worst. On Rosenbrock and Rastrigin functions, all algorithms didn't performed well. However, the results of IBFO are a little better than that of PSO. On Ackley function, BFO, PSO and GA all performed badly while only IBFO obtained remarkable results. IBFO converged fast at the end stage and seemed was able to continue improving its result. On Griewank function, it got a rank of 2. However, it is only a little worse than PSO. On Schwefel2.22, it performed better than the other three algorithms, too. Overall, IBFO shows significant improvement over the original BFO algorithm. And its optimization ability is better than the classic PSO and GA algorithms on most functions.



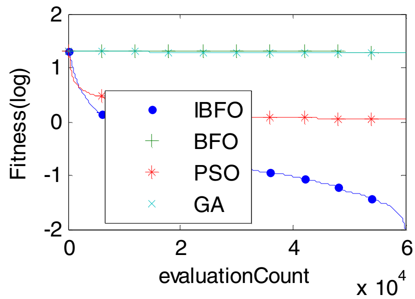
(a) Sphere



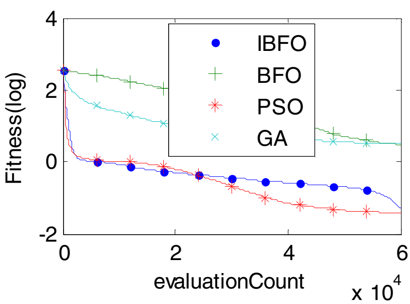
(b) Rosenbrock



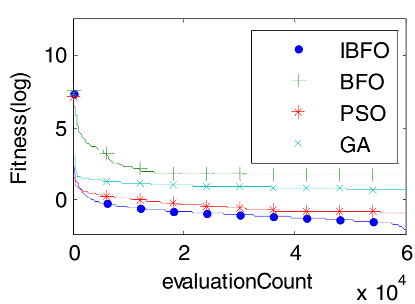
(c) Rastrigin



(d) Ackley



(e) Griewank



(f) Schwefel2.22

Fig. 1. Convergence plots of IBFO, BFO, PSO and GA algorithms on functions

Table 2. Results obtained by the IBFO, BFO, PSO and GA algorithms

Function		IBFO	BFO	PSO	GA
f_1	Mean	**3.16004e-007**	6.17635e-001	4.52322e-006	6.61625e-001
	Std	**1.50001e-007**	1.94300e-001	3.34529e-006	2.34433e-001
f_2	Mean	**2.05631e+001**	2.02447e+003	2.49117e+001	6.56345e+002
	Std	**1.71408e+001**	8.62087e+002	2.12081e+001	4.55687e+002
f_3	Mean	**1.32458e+001**	4.94922e+002	3.35838e+001	6.80152e+001
	Std	**5.41958e+000**	6.74151e+001	1.07949e+001	1.77908e+001
f_4	Mean	**1.00880e-002**	1.95483e+001	1.10071e+000	1.86919e+001
	Std	**1.47724e-003**	3.71818e-001	9.14650e-001	1.30767e-000
f_5	Mean	4.96128e-002	3.00705e+000	**3.89587e-002**	3.11937e+000
	Std	3.97896e-002	4.5062e-001	**3.13277e-002**	8.55760e-001
f_6	Mean	**8.51024e-003**	5.43634e+001	1.39941e-001	4.92696e+000
	Std	**1.71202e-003**	1.20626e+001	1.87068e-001	1.09457e+000

5 Conclusions

This paper analyzes the shortages of original BFO algorithm. To overcome its shortages, an Improved Bacterial Foraging Optimization (IBFO) algorithm is proposed. Social cooperation and adaptive step size strategies are used in IBFO. In the chemotactic steps, the tumble angles are no longer generated randomly. Instead, they are produced using the information of the bacteria's global best, the bacterium' personal best and its current position. The step size in chemotactic processes decreases linearly with iterations too, which could balance its exploration ability and exploitation ability.

To test the optimization ability of IBFO algorithm, it is tested on a set of benchmark functions compared with original BFO, PSO and GA algorithms. The results show that IBFO algorithm performed best on five functions of all six. On the rest one, it is only a little worse than PSO. In general, the proposed IBFO algorithm offers significant improvements over original BFO, and is a competitive algorithm for optimization compared other algorithms.

References

1. Passino, K.M.: Biomimicry of Bacterial Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine 22, 52–67 (2002)
2. Chen, H., Zhu, Y., Hu, K.: Cooperative Bacterial Foraging Optimization. Discrete Dynamics in Nature and Society, Article ID 815247, 17 pages (2009)

3. Wu, C., Zhang, N., Jiang, J., Yang, J., Liang, Y.: Improved Bacterial Foraging Algorithms and Their Applications to Job Shop Scheduling Problems. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007, Part I. LNCS, vol. 4431, pp. 562–569. Springer, Heidelberg (2007)
4. Majhi, R., Panda, G., Majhi, B., Sahoo, G.: Efficient Prediction of Stock Market Indices Using Adaptive Bacterial Foraging Optimization (ABFO) and BFO Based Techniques. Expert Systems with Applications 36(6), 10097–10104 (2009)
5. Eberhart, R.C., Shi, Y., Kennedy, J.: Swarm Intelligence. Morgan Kaufmann (2001)
6. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, pp. 303–308 (1998)
7. Chen, H., Zhu, Y., Hu, K.: Self-Adaptation in Bacterial Foraging Optimization Algorithm. In: Proceedings of the 3rd International Conference on Intelligent System & Knowledge Engineering, Xiamen, China, pp. 1026–1031 (2008)
8. Zhou, B., Gao, L., Dai, Y.: Gradient Methods with Adaptive Step-Sizes. Computational Optimization and Applications 35(1), 69–86 (2006)
9. Karaboga, D., Basturk, B.: A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm. Journal of Global Optimization 39(3), 459–471 (2007)
10. Zou, W., Zhu, Y., Chen, H., Sui, X.: A Clustering Approach Using Cooperative Artificial Bee Colony Algorithm. Discrete Dynamics in Nature and Society, Article ID 459796, 16 pages (2010)
11. Yan, X., Zhu, Y., Zou, W.: A Hybrid Artificial Bee Colony Algorithm for Numerical Function Optimization. In: 11th International Conference on Hybrid Intelligent Systems, pp. 127–132 (2011)
12. Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S.: Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. IEEE Transcations on Evolutionary Computing 10, 281–295 (2006)
13. Shi, Y., Eberhart, R.C.: Empirical Study of Particle Swarm Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, Piscataway, NJ, USA, vol. 3, pp. 1945–1950 (1999)