Hindawi Publishing Corporation
Applied Computational Intelligence and Soft Computing
Volume 2012, Article ID 907853, 7 pages
doi:10.1155/2012/907853

Research Article

A Crossover Bacterial Foraging Optimization Algorithm

Rutuparna Panda and Manoj Kumar Naik

Department of Electronics and Telecommunication Engineering, VSS University of Technology, Burla 768018, India

Correspondence should be addressed to Rutuparna Panda, r_ppanda@yahoo.co.in

Received 16 April 2012; Revised 11 July 2012; Accepted 9 August 2012

Academic Editor: Jun He

This paper presents a modified bacterial foraging optimization algorithm called crossover bacterial foraging optimization algorithm, which inherits the crossover technique of genetic algorithm. This can be used for improvising the evaluation of optimal objective function values. The idea of using crossover mechanism is to search nearby locations by offspring (50 percent of bacteria), because they are randomly produced at different locations. In the traditional bacterial foraging optimization algorithm, search starts from the same locations (50 percent of bacteria are replicated) which is not desirable. Seven different benchmark functions are considered for performance evaluation. Also, comparison with the results of previous methods is presented to reveal the effectiveness of the proposed algorithm.

1. Introduction

Nowadays several algorithms are developed that are inspired by the nature. The main principle behind the nature-inspired algorithm is interpreted as the capacity of an individual to obtain sufficient energy source in the least amount of time. In the process of foraging, the animals with poor foraging strategies are eliminated, and successful ones tend to propagate [1]. One of the most successful foragers is *E. coli* bacteria (those living in our intestines), which use chemical sensing organs to detect the concentration of nutritive and noxious substances in its environment. The bacteria then move within the environments via tumble and runs, avoiding the noxious substances and getting closer to food patch areas in a process called chemotaxis. Based on the *E. coli* foraging strategy, Passino proposed bacterial foraging optimization algorithm (BFOA) [2–4] which maximizes the energy intake per unit time. So as to improve BFOA performance, a large number of modifications have already been undertaken. Some of the modifications are directly based on analysis of the components [5–8] while others are named as hybrid algorithms [9–11].

During the past two decades, the genetic algorithm (GA) has claimed its suitability for dealing with optimization problems by academic and industrial communities. A possible solution to a specific problem is encoded as a chromosome, which consists of a group of genes. Each chromosome refers to a search space and is decided by a fitness evaluation. The GA uses basic genetic operators such as crossover and mutation to produce the genetic composition of a population. The crossover operator produces two offspring by recombining the information of two parents. Randomly gene values are changed using the mutation operator. The crossover and mutation applicability is determined by the crossover probability and mutation probability [12].

In this paper, we present some modifications for the BFOA by adapting the crossover operator used in GA. Here 50 percent of healthier bacteria are used for crossover with some crossover probability to produce 50 percent of bacteria as offspring. These offspring bacteria are produced at different locations and start searching. But in BFOA, 50 percent of bacteria are replicated at the same location and start searching from the same location. As a result they miss some useful parameters in the search space. This has motivated us to investigate crossover BFOA, which can find global optimal solution more effectively. The paper is organised as follow. In Section 2, we describe the bacterial foraging optimization algorithm. Section 3 presents the proposed modification to the BFOA. Section 4 deals with the comparison of the proposed algorithm CBFOA with BFOA,

Adaptive BFOA (ABFOA) [8], and genetic algorithm (GA) [12] using some common benchmark functions. Finally, conclusion and future scope of the work are presented in Section 5.

2. The Bacterial Foraging Optimization Algorithm

Suppose that we want to find the minimum of $J(\theta)$, $\theta \in \Re^p$, where we do not have measurements or an analytical description of the gradient $\nabla J(\theta)$. Here, we use BFOA to solve this nongradient optimization problem. Let θ be the position of the bacterium and let $J(\theta)$ represent the cost of the optimization problem, with $J(\theta) < 0$, $J(\theta) = 0$, and $J(\theta) > 0$. These values guide us about the bacterium location (whether in nutrient-rich, neutral, or noxious environments). So basically the BFOA consists of four principal mechanisms known as chemotaxis, swarming, reproduction, and elimination-dispersal.

2.1. Chemotaxis. The process simulates the movement of the bacteria via swimming and tumbling. Let $J(i, j, k, l)$ denote the cost at the location of the ith bacterium, and let $\theta^i(j, k, l)$ represent jth chemotactic, kth reproduction, and lth elimination-dispersal events. Let C steps (during runs) be taken in the random direction specified by the tumble. Then the chemotactic movement can be represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}, \tag{1}$$

where $\Delta(i)$ is a random vector with each elements lying in $[-1, 1]$.

2.2. Swarming. During the movements, cells release attractants and repellents to signal other cells so that they should swarm together, provided that they get nutrient-rich environment or avoided the noxious environment. The cell-to-cell attraction and repelling effects are denoted as

$$\begin{aligned} &J_{cc}(\theta, P(j,k,l)) \\ &= \sum_{i=1}^{S} J_{cc}^i\left(\theta, \theta^i(j,k,l)\right) \\ &= \sum_{i=1}^{S}\left[-d_{\text{attract}}\exp\left(-w_{\text{attract}}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right] \\ &\quad + \sum_{i=1}^{S}\left[-h_{\text{repellant}}\exp\left(-w_{\text{repellant}}\sum_{m=1}^{p}\left(\theta_m - \theta_m^i\right)^2\right)\right], \end{aligned} \tag{2}$$

where $J_{cc}(\theta, P(j, k, l))$ is the objective function value to be added to the actual objective function to present time varying objective function, S is the total number of bacteria, p is the number of variables involved in the search space, $\theta = [\theta_1, \theta_2, \ldots, \theta_p]^T$ is a point on the optimization domain, and θ_m^i is the mth components of the ith bacterium position θ^i. d_{attract}, w_{attract}, $h_{\text{repellant}}$, and $w_{\text{repellant}}$ are different coefficients used for signalling.

2.3. Reproduction. The population is sorted in ascending order of accumulated cost, then $S_r(= S/2)$ least healthy bacteria die and the other $S_r(= S/2)$ healthiest bacteria are considered for reproduction, each split into two bacteria, which are placed at the same location. This allows us to keep a constant population size, which is convenient in coding the algorithm.

2.4. Elimination Dispersal. Due to gradual or sudden change in the local environment, the life of the bacteria may be affected. So in order to incorporate this phenomenon, we eliminate each bacterium in the population with the probability p_{ed} and a new replacement is randomly initialized over the search space.

3. The Crossover Bacterial Foraging Optimization Algorithm

The main aim of the CBFOA is to find the minimum of a function $J(\theta)$, $\theta \in \Re^p$, which is not in the gradient $\nabla J(\theta)$. Here $J(\theta)$ is an attractant-repellent profile and θ is the position of a bacterium. Let $P(j, k, l) = \{\theta^i(j, k, l) \mid i = 1, 2, \ldots, S\}$ represent the position of each bacterium in the population of S bacterium at the jth chemotactic step, kth crossover-reproduction step, and lth elimination-dispersal events. Here, let $J(i, j, k, l)$ denote the cost at the location of the ith bacterium at position $\theta^i(j, k, l) \in \Re^p$. Let $C(i) > 0$ the step size taken in the random direction represent a tumble. Note that the position of bacterium for the next chemotactic steps will be

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\varnothing(i). \tag{3}$$

If at the location $\theta^i(j + 1, k, l)$ the cost $J(i, j + 1, k, l)$ is better (lower) than the location at $\theta^i(j, k, l)$, then another step of size $C(i)$ (in the same direction) will be taken. This swim is continued as long as it reduces the cost, but it is allowed only up to a maximum number of steps N_s. We have to implement in such a manner that the bacterium can swarm together via an attractant and repellent, cell-to-cell signalling effect as modelled in (2). We consider the swarming effects, the ith bacterium, $i = 1, 2, \ldots, S$, will hill climb on

$$J(i, j, k, l) + J_{cc}(\theta, P), \tag{4}$$

so that the cells will try to find nutrients, avoid noxious substances, and at the same time try to move towards other cells, but not too close to them.

After the N_c chemotactic steps, a crossover-reproduction step is taken. Let N_{cr} be the number of crossover-reproduction steps to be taken. After the chemotactic steps, the population is going to reproduce for the next generation which consists of sufficient nutrients. For the convenience, we consider S to be a positive number (divisible by (4)). Let

$$S_c = \frac{S}{2} \tag{5}$$

be the number (population) having sufficient nutrients, which can go for next generation. For the crossover-reproduction steps, the population is sorted in order of ascending accumulated cost (higher cost means the nutrient value is less); then the S_c least healthy bacteria die and the other S_c healthiest bacteria have gone through the crossover with the probability p_c to get S_c child bacteria. Then the new set of bacteria can be formed by appending the S_c number healthiest (parent) bacteria and S_c number of child bacteria. This helps that search domain is more dynamic in nature as parent bacteria start search in the next generation where nutrient concentration is more and child bacterium searches its nearby place that may be untouched due to using the BFOA search strategy.

Let N_{ed} be the number of elimination-dispersal events, and for each elimination-dispersal event each bacterium in the population is subjected to eliminate dispersal with probability p_{ed}. This helps to keep track of sudden change in the environmental condition, which may affect life of the bacterium, so new set of bacterium can be introduced in the search domain.

3.1. Crossover Bacterial Foraging Optimization Algorithm. First initialize the parameters p, S, N_c, N_s, N_{cr}, N_{ed}, p_c, p_{ed}, and $C(i)$, where p represent dimension of search space, S represent the number of bacterium involved in the population, N_c represent the number of chemotactic steps, N_s represent the maximum swim length, N_{cr} represent the number of crossover-reproduction steps, N_{ed} represent the number of elimination-dispersal steps, p_c represent the probability of crossover, p_{ed} represent the probability of elimination dispersal event, and $C(i)$ is size of the step taken in the random direction specified by a tumble. If we use the swarming, we have to pick the parameters of the cell-to-cell attractant as d_{attract} (depth of the attractant by the cell), w_{attract} (width of the attractant signals), $h_{\text{repellant}}$ (magnitude of the height repellent effect), and $w_{\text{repellant}}$ (magnitude of the width repellent effect). We have also initialized θ^i, $i = 1, 2, \ldots, S$ randomly within the search space. This algorithm also modelled bacterial population chemotaxis, elimination, and dispersal steps as reported by Passino [2] and explained in Section 2. In this paper, what is new is the reproduction step. Instead of using the procedure for reproduction explained in Section 2, here a new idea of crossover-reproduction is introduced:

(initially, $j = k = l = 0$).

Step 1. Elimination-dispersal loop: $l = l + 1$.

Step 2. Crossover-reproduction loop: $k = k + 1$.

Step 3. Chemotaxis loop: $j = j + 1$.

(a) For $i = 1, 2, \ldots, S$ take a chemotactic step for bacterium i as follows.

(b) Compute cost function $J(i, j, k, l)$.

(c) Then compute $J(i, j, k, l) = J(i, j, k, l) + J(\theta, P)$ (i.e., add on the cell-to-cell signalling effects).

(d) Let $J_{\text{last}} = J(i, j, k, l)$ to save this value since we may find a better cost via a run.

(e) Tumble: generate a random vector $\Delta(i) \in \Re^p$ with each element $\Delta_m(i)$, $m = 1, 2, \ldots, p$, a random number on $[-1, 1]$.

(f) Move: let $\theta^i(j + 1, k, l) = \theta^i(j, k, l) + C(i)(\Delta(i)/\sqrt{\Delta^T(i)\Delta(i)})$. This results in step size taken in the direction of tumble of the ith bacterium.

(g) Then compute $J(i, j + 1, k, l)$ and let $J(i, j + 1, k, l) = J(i, j + 1, k, l) + J_{cc}(\theta^i(j + 1, k, l), P(j + 1, k, l))$.

(h) Swim:

(i) Let $m = 0$ (counter for swim length).

(ii) While $m < N_s$ (it have not climbed down too long),

(1) let $m = m + 1$;

(2) if $J(i, j + 1, k, l) < J_{\text{last}}$ (if doing better), let $J_{\text{last}} = J(i, j + 1, k, l)$ and let $\theta^i(j + 1, k, l) = \theta^i(j + 1, k, l) + C(i)(\Delta(i)/\sqrt{\Delta^T(i)\Delta(i)})$ and use this $\theta^i(j + 1, k, l)$ to compute the new $J(i, j + 1, k, l)$ as we did in (g);

(3) else, let $m = N_s$, come out from the while loop;

(iii) go to next bacterium $(i + 1)$ if $i \neq S$, then go to (b) to process the next bacterium.

Step 4. If $j < N_c$, go to Step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.

Step 5. Crossover reproduction.

(a) For the given k and l, and for each $i = 1, 2, \ldots, S$, let $J^i_{\text{health}} = \sum_{j=1}^{N_c+1} J(i, j, k, l)$ be the health of bacterium i (a measure of how many nutrients it got over its lifetime and how successful it was at avoiding noxious substances). Sort bacteria and chemotactic parameter $C(i)$ in ascending order of cost J_{health} (higher cost means lower health).

(b) The S_c bacteria with the highest J_{health} values die and the other S_c bacteria are treated as parent bacterium for the next generation.

(c) Then we choose two sets of parent bacterium from the S_c healthiest bacteria and crossover them with probability p_c to get S_c number of offspring bacterium.

(d) Then append the S_c number of parent (healthiest) bacterium and S_c number of offspring bacterium to form complete set of S bacterium.

Step 6. If $k < N_{\text{cr}}$, go to Step 2. In this case we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

Step 7. Elimination dispersal: for $i = 1, 2, \ldots, S$, with probability p_{ed}, eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population

Table 1: Description of benchmark functions used.

Function	Mathematical representation	Range of search	Theoretical optima
Ackley	$f_1(\vec{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right) + 20 + e$	$(-5,5)^n$	$f_1(\vec{0}) = 0$
Griewank	$f_2(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n}x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$(-10,10)^n$	$f_2(\vec{0}) = 0$
Rastrigin	$f_3(\vec{x}) = \sum_{i=1}^{n}[x_i^2 - 10\cos 2\pi x_i + 10]$	$(-5,5)^n$	$f_3(\vec{0}) = 0$
Rosenbrock	$f_4(\vec{x}) = \sum_{i=1}^{n-1}\left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]$	$(-2,2)^n$	$f_4(\vec{0}) = 0$
Rotated hyperellipsoid	$f_5(\vec{x}) = \sum_{i=1}^{n}\sum_{j=1}^{i}x_j^2$	$(-5,5)^n$	$f_5(\vec{0}) = 0$
De Jong's	$f_6(\vec{x}) = \sum_{i=1}^{n}x_i^2$	$(-5,5)^n$	$f_6(\vec{0}) = 0$
Weighted sphere model	$f_7(\vec{x}) = \sum_{i=1}^{n}(i \cdot x_i^2)$	$(-5,5)^n$	$f_6(\vec{0}) = 0$

Table 2: Parameters used for the benchmark.

Algorithm	Parameters
CBFOA	$N_c = 50$, $N_{cr} = 20$, $N_s = 3$, $p_c = 0.7$, $C(i) = 0.01 \times$ Range, $d_{attract} = 0.001$, $w_{attract} = 0.02$, $h_{repellant} = 0.001$, $w_{repellant} = 10$
BFOA	$N_c = 50$, $N_{cr} = 20$, $N_s = 3$, $C(i) = 0.01 \times$ Range, $d_{attract} = 0.001$, $w_{attract} = 0.02$, $h_{repellant} = 0.001$, $w_{repellant} = 10$
ABFOA	$N_c = 50$, $N_{cr} = 20$, $N_s = 3$, $d_{attract} = 0.001$, $w_{attract} = 0.02$, $h_{repellant} = 0.001$, $w_{repellant} = 10$, $\lambda = 4000$
GA	$p_c = 0.7$, $p_m = 0.3$

constant. To do this, if we eliminate a bacterium, simply disperse one to a random location on the optimization domain.

Step 8. If $l < N_{re}$, then go to Step 1; otherwise end.

4. Experimental Results

This section illustrates some comparisons between the proposed CBFOA, BFOA [2], adaptive BFOA [8], and GA [12] using some numerical benchmark test functions described in Table 1.

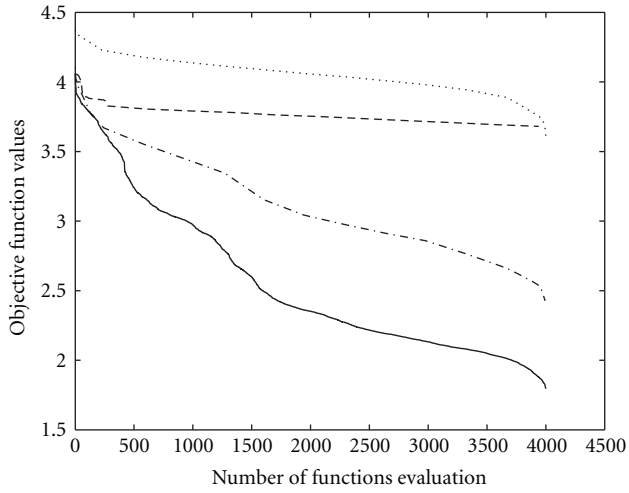
The search dimensions for all test problems we consider here are 50 and 500 for comparing algorithm performance. We also choose two variants in the bacterial or gene population, one with $S = 4$ and other with $S = 20$. Here we take the crossover rate fixed and implement uniform crossover. For the simplicity of the algorithm, we neglect the elimination and dispersal event. Note that the parameters considered for the algorithms are given in Table 2. We take results for 100 independent runs and report the minimum, the mean, and the standard deviation of the final objective function values for all four algorithms. These results are shown in Table 3. Finally, the performances of all four different algorithms are illustrated in Figure 1.

5. Conclusion

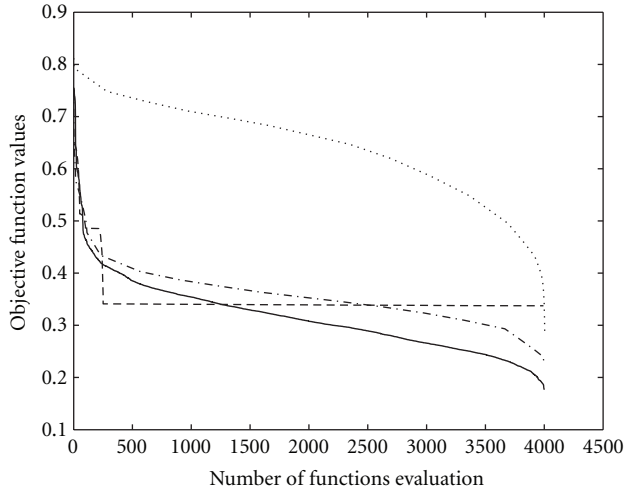
Recently, many modified bacterial foraging optimization algorithms have been investigated for improving the learning and speed for convergence. Research is more or less

Table 3: Minimum, mean values, and standard deviation for the benchmark function for f_1–f_7 (represent up to three fractional points).

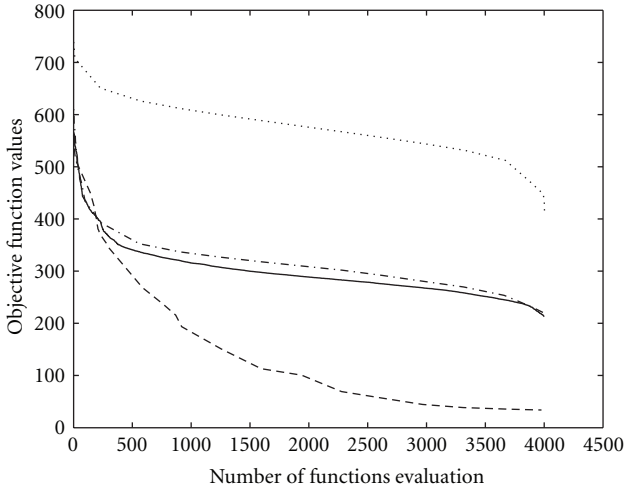
Function	Statistical measurement	CBFOA			BFOA		
		$p = 50, S = 4$	$p = 50, S = 20$	$p = 500, S = 4$	$p = 50, S = 4$	$p = 50, S = 20$	$p = 500, S = 4$
	Minimum	1.525	1.319	3.262	1.715	1.456	3.305
f_1	Mean	2.160	1.613	3.392	2.267	1.741	3.437
	Std. Dev.	0.225	0.113	0.048	0.270	0.146	0.052
	Minimum	0.136	0.134	0.421	0.151	0.136	0.449
f_2	Mean	0.201	0.166	0.485	0.208	0.172	0.498
	Std. Dev.	0.028	0.013	0.021	0.023	0.016	0.019
	Minimum	164.455	152.077	2712.102	172.454	154.242	2895.302
f_3	Mean	197.770	175.040	2908.327	202.866	183.931	2980.910
	Std. Dev.	15.348	11.728	86.705	15.732	11.577	85.742
	Minimum	73.536	68.936	5233.215	74.104	70.381	5312.160
f_4	Mean	172.978	112.322	5641.901	163.977	104.736	5806.152
	Std. Dev.	44.929	35.153	218.398	41.819	26.631	219.462
	Minimum	0.932	0.811	75.667	0.935	0.850	78.766
f_5	Mean	1.302	1.054	88.812	1.363	1.106	91.327
	Std. Dev.	0.193	0.095	4.865	0.195	0.113	4.838
	Minimum	0.838	0.829	75.353	0.998	0.892	81.900
f_6	Mean	1.319	1.053	88.355	1.366	1.096	91.500
	Std. Dev.	0.193	0.091	5.161	0.203	0.111	4.493
	Minimum	23.091	19.592	15919.004	23.241	20.682	18340.260
f_7	Mean	33.173	25.865	19284.121	33.991	26.904	20135.431
	Std. Dev.	4.810	2.804	1232.30	6.039	3.076	1050.504
Function	Statistical measurement	ABFOA			GA		
		$p = 50, S = 4$	$p = 50, S = 20$	$p = 500, S = 4$	$p = 50, S = 4$	$p = 50, S = 20$	$p = 500, S = 4$
	Minimum	3.274	3.182	3.756	3.253	3.334	3.734
f_1	Mean	3.620	3.450	3.837	3.529	3.521	3.797
	Std. Dev.	0.120	0.089	0.030	0.063	0.054	0.015
	Minimum	0.299	0.256	0.542	0.248	0.225	0.512
f_2	Mean	0.428	0.363	0.648	0.299	0.273	0.560
	Std. Dev.	0.059	0.039	0.040	0.020	0.019	0.014
	Minimum	16.401	16.526	4265.352	337.504	354.976	4616.801
f_3	Mean	27.678	24.500	4559.012	405.755	401.736	4820.149
	Std. Dev.	5.028	3.219	74.756	18.318	15.624	53.202
	Minimum	52.703	49.708	8809.955	542.544	486.038	8413.197
f_4	Mean	669.706	233.259	9583.201	615.075	571.955	8814.362
	Std. Dev.	217.473	237.100	318.864	35.159	31.871	139.747
	Minimum	7.596	6.305	109.191	10.536	9.701	143.726
f_5	Mean	10.175	8.998	118.319	11.954	11.874	151.972
	Std. Dev.	0.949	0.665	4.176	0.593	0.777	2.242
	Minimum	8.270	7.543	104.974	10.350	9.736	142.649
f_6	Mean	10.181	9.086	118.443	12.179	11.957	152.154
	Std. Dev.	0.792	0.576	3.774	0.635	0.684	2.076
	Minimum	12.845	11.466	37978.059	230.363	233.034	35713.922
f_7	Mean	18.035	14.753	39957.690	288.583	282.761	37448.482
	Std. Dev.	2.381	1.197	1189.191	20.812	20.009	641.561



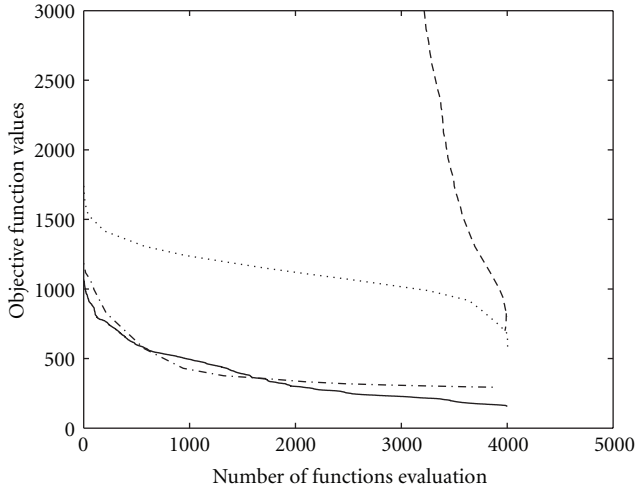
(a) Ackley function figure



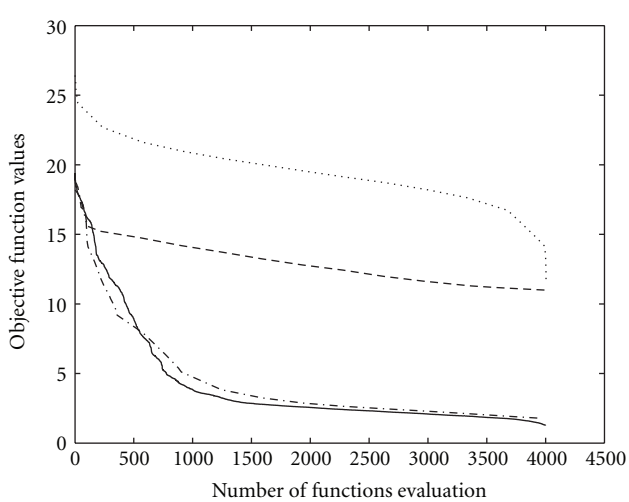
(b) Griewank function



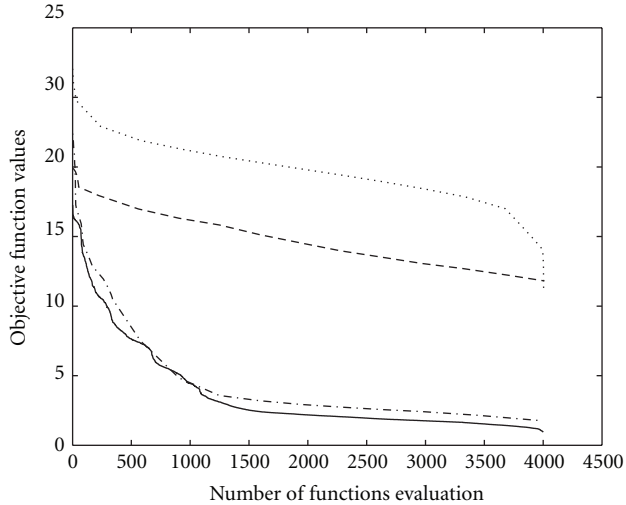
(c) Rastrigin function figure



(d) Rosenbrock function

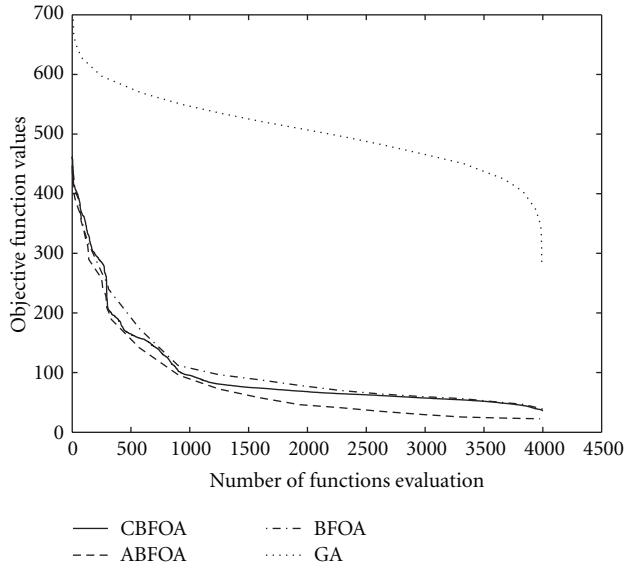


(e) Rotated hyperellipsoid function



(f) De Jong's function

Figure 1: Continued.



(g) Weighted sphere model function

Figure 1: Performance of the CBFOA, ABFOA, BFOA and GA for the various functions for dimension = 50.

concentrated to get local optimal or suboptimal solutions. However, this paper proposes a modified bacteria foraging optimization algorithm for finding global optimal solutions with adapted crossover properties of genetic algorithm. The performance of the proposed algorithm is illustrated by taking various benchmark test functions. From the numerical results, it is evident that the proposed CBFOA outperforms ABFOA, BFOA, and GA reported earlier. The proposed algorithm has potential and can be used in various optimization problems, where social foraging model works.

References

[1] 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, no. 3, pp. 603–628, 2002.

[2] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control," *IEEE Control Systems Magazine*, vol. 22, no. 3, pp. 52–67, 2002.

[3] V. Gazi and K. M. Passino, "Stability analysis of swarms in an environment with an attractant/repellent profile," in *Proceedings of the American Control Conference*, pp. 1819–1824, Anchorage, Alaska, USA, May 2002.

[4] V. Gazi and K. M. Passino, "Stability analysis of swarms," in *Proceedings of the American Control Conference*, pp. 1813–1818, Anchorage, Alaska, USA, May 2002.

[5] A. Abraham, A. Biswas, S. Dasgupta, and S. Das, "Analysis of reproduction operator in Bacterial Foraging Optimization Algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1476–1483, June 2008.

[6] A. Biswas, S. Das, S. Dasgupta, and A. Abraham, "Stability analysis of the reproduction operator in bacterial foraging optimization," in *Proceedings of the 5th International Conference on Soft Computing As Transdisciplinary Science and Technology (CSTST '08)*, pp. 564–571, ACM, New York, NY, USA, October 2008.

[7] S. Das, S. Dasgupta, A. Biswas, A. Abraham, and A. Konar, "On stability of the chemotactic dynamics in bacterial-foraging optimization algorithm," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 39, no. 3, pp. 670–679, 2009.

[8] S. Dasgupta, S. Das, A. Abraham, and A. Biswas, "Adaptive computational chemotaxis in bacterial foraging optimization: an analysis," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 919–941, 2009.

[9] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "A synergy of differential evolution and bacterial foraging optimization for global optimization," *Neural Network World*, vol. 17, no. 6, pp. 607–626, 2007.

[10] D. H. Kim, A. Abraham, and J. H. Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization," *Information Sciences*, vol. 177, no. 18, pp. 3918–3937, 2007.

[11] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "Synergy of PSO and bacterial foraging optimization—a comparative study on numerical benchmarks," *Advances in Soft Computing*, vol. 44, pp. 255–263, 2007.

[12] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1996.