Synergy of PSO and Bacterial Foraging Optimization – A Comparative Study on Numerical Benchmarks

Arijit Biswas[1], Sambarta Dasgupta[1], Swagatam Das[1], and Ajith Abraham[2]

[1] Dept. of Electronics and Telecommunication Engg,
Jadavpur University, Kolkata, India
[2] Norwegian University of Science and Technology, Norway
arijitbiswas87@gmail.com, sambartadg@gmail.com,
swagatamdas19@yahoo.co.in, ajith.abraham@ieee.org

Abstract. Social foraging behavior of *Escherichia coli* bacteria has recently been explored to develop a novel algorithm for distributed optimization and control. The Bacterial Foraging Optimization Algorithm (BFOA), as it is called now, is currently gaining popularity in the community of researchers, for its effectiveness in solving certain difficult real-world optimization problems. Until now, very little research work has been undertaken to improve the convergence speed and accuracy of the basic BFOA over multi-modal fitness landscapes. This article comes up with a hybrid approach involving Particle Swarm Optimization (PSO) and BFOA algorithm for optimizing multi-modal and high dimensional functions. The proposed hybrid algorithm has been extensively compared with the original BFOA algorithm, the classical g_best PSO algorithm and a state of the art version of the PSO. The new method is shown to be statistically significantly better on a five-function test-bed and one difficult engineering optimization problem of spread spectrum radar poly-phase code design.

Keywords: Bacterial Foraging, hybrid optimization, particle swarm optimization, Radar poly-phase code design.

1 Introduction

In 2001, Prof. K. M. Passino proposed an optimization technique known as Bacterial Foraging Optimization Algorithm (BFOA) based on the foraging strategies of the E. Coli bacterium cells [1]. Until date there have been a few successful applications of the said algorithm in optimal control engineering, harmonic estimation [2], transmission loss reduction [3], machine learning [4] and so on. Experimentation with several benchmark functions reveal that BFOA possesses a poor convergence behavior over multi-modal and rough fitness landscapes as compared to other naturally inspired optimization techniques like the Genetic Algorithm (GA) [5] Particle Swarm Optimization (PSO) [6] and Differential Evolution (DE)[7]. Its performance is also heavily affected with the growth of search space dimensionality. In 2007, Kim *et al.* proposed a hybrid approach involving GA and BFOA for function optimization [8]. The proposed algorithm outperformed both GA and BFOA over a few numerical benchmarks and a practical PID tuner design problem.

E. Corchado et al. (Eds.): Innovations in Hybrid Intelligent Systems, ASC 44, pp. 255–263, 2007.
springerlink.com

In this article we come up with a hybrid optimization technique, which synergistically couples the BFOA with the PSO. The later is a very popular optimization algorithm these days and it draws inspiration from the group behavior of a bird flock or school of fish etc. The proposed algorithm performs local search through the chemotactic movement operation of BFOA whereas the global search over the entire search space is accomplished by a PSO operator. In this way it balances between *exploration and exploitation* enjoying best of both the worlds.

The proposed algorithm, referred to as Bacterial Swarm Optimization (BSO) has been extensively compared with the classical PSO, a state-of- the-art variant of PSO and the original BFOA over a test suit of five well-known benchmark functions and also on a practical optimization problem of spread spectrum radar poly-phase code design [9]. The following performance metrics were used in the comparative study (i) quality of the final solution, (ii) convergence speed, (iii) robustness and (iv) scalability. Such comparison reflects the superiority of the proposed approach.

2 The Bacterial Swarm Optimization Algorithm

Particle swarm optimization (PSO) [6] is a stochastic optimization technique that draws inspiration from the behavior of a flock of birds or the collective intelligence of a group of social insects with limited individual capabilities. In PSO a population of particles is initialized with random positions $\vec{X}_i$ and velocities $\vec{V}_i$, and a fitness function, *f*, is evaluated, using the particle's positional coordinates as input values. In an n-dimensional search space, $\vec{X}_i = (x_{i1}, x_{i2}, x_{i3},...,x_{in})$ and $\vec{V}_i = (v_{i1}, v_{i2}, v_{i3},...,v_{in})$. Positions and velocities are adjusted, and the function is evaluated with the new coordinates at each time-step. The velocity and position update equations for the d-th dimension of the i-th particle in the swarm may be given as follows:

$$\left.\begin{aligned} V_{id}(t+1) &= \omega.V_{id}(t) + C_1.\varphi_1.(P_{lid} - X_{id}(t)) + C_2.\varphi_2.(P_{gd} - X_{id}(t)) \\ X_{id}(t+1) &= X_{id}(t) + V_{id}(t+1) \end{aligned}\right\} \quad (1)$$

The BFOA is on the other hand is based upon search and optimal foraging decision making capabilities of the *E.Coli* bacteria [10]. The coordinates of a bacterium here represent an individual solution of the optimization problem. Such a set of trial solutions converges towards the optimal solution following the foraging group dynamics of the bacteria population. Chemo-tactic movement is continued until a bacterium goes in the direction of positive nutrient gradient (i. e. increasing fitness). After a certain number of complete swims the best half of the population undergoes reproduction, eliminating the rest of the population. In order to escape local optima, an elimination-dispersion event is carried out where, some bacteria are liquidated at random with a very small probability and the new replacements are initialized at random locations of the search space. A detailed description of the complete algorithm can be traced in [1].

In the proposed approach, after undergoing a chemo-tactic step, each bacterium also gets mutated by a PSO operator. In this phase, the bacterium is stochastically attracted towards the globally best position found so far in the entire population at

current time and also towards its previous heading direction. The PSO operator uses only the 'social' component and eliminates the 'cognitive' component as the local search in different regions of the search space is already taken care of by the chemo-tactic steps of the BFOA algorithm. In what follows we briefly outline the new BSO algorithm step by step.

[Step 1] Initialize parameters n, N, N_C, N_S, N_{re}, N_{ed}, P_{ed}, $C(i)$ $(i=1,2,\ldots,N)$, ϕ^i.

Where,

n: Dimension of the search space,

N: The number of bacteria in the population,

N_C: No. of Chemo-tactic steps,

N_{re}: The number of reproduction steps,

N_{ed}: The number of elimination-dispersal events,

P_{ed}: Elimination-dispersal with probability,

$C(i)$: The size of the step taken in the random direction specified by the tumble.

ω: The inertia weight.

C_1: Swarm Confidence.

$\vec{\theta}(i, j, k)$: Position vector of the i-th bacterium, in j-th chemotactic step, and k-th reproduction.

$\vec{V}_i$: Velocity vector of the i-th bacterium.

[Step 2] Update the following:

J (i, j, k): Cost or fitness value of the i-th bacterium in the jth chemo-taxis, and k-th reproduction loop.

$\vec{\theta}_{g\_best}$: Position vector of the best position found by all bacteria.

J_{best} (i, j, k): Fitness of the best position found so far.

[Step 3] Reproduction loop: $k=k+1$

[Step 4] Chemotaxis loop: $j=j+1$

[substep a] For $i=1,2,\ldots,N$, take a chemotactic step for bacterium i as follows.

[substep b] Compute fitness function, J (i, j, k).

[substep c] Let $J_{last}=J(i,j,k)$ to save this value since we may find a better cost via a run.

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

[substep e] Move: Let $\theta(i, j+1, k) = \theta(i, j, k) + C(i)\dfrac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$

[substep f] Compute $J(i, j+1, k)$.

[substep g] Swim: we consider only the i-th bacterium is swimming while the others are not moving then.

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

ii) While m< N_s (if have not climbed down too long).

- Let m=m+1.
- If $J(i, j+1, k) < J_{last}$ (if doing better), let $J_{last} = J(i, j+1, k)$ and let

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

and use this $\theta(i, j+1, k)$ to compute the new $J(i, j+1, k)$ as we did in [sub step f]

- Else, let m= N_s . This is the end of the while statement.

[Substep 5] Mutation with PSO Operator

For i = 1,2, …, S

- Update the $\vec{\theta}_{g\_best}$ and J_{best} (i, j, k)
- Update position and velocity of the d-th coordinate of the i-th bacterium according to the following rule:

$$V_{id}^{new} = \omega.V_{id}^{new} + C_1.\varphi_1.(\theta_{g\_best_d} - \theta_d^{old}(i, j+1, k))$$

$$\theta_d^{new}(i, j+1, k) = \theta_d^{old}(i, j+1, k) + V_{id}^{new}$$

[Step 6] Let S_r=S/2.

The S_r bacteria with highest cost function (J) values die and the other half of bacteria population with the best values split (and the copies that are made are placed at the same location as their parent).

[Step 7] If k<N_{re}, go to step 1. We have not reached the specified number of reproduction steps. So we start the next generation in the chemo-taxis loop.

3 Experimental Setup

3.1 Benchmark Functions Used

The performance of the BSO algorithm has been evaluated on a test bed of 5 well-known benchmark functions [10] as shown in table 1. In table 1 n represents the number of dimensions and we used n=15, 30, 45 and 60. All the benchmark functions except f_5 have their global minima at the origin or very near to the origin. For Shekel's Foxholes the global minimum is at (-31.95,-31.95) and its value is 0.998. An asymmetrical initialization procedure has been used here following the work reported in [12]. A famous NP-hard problem of optimal design arises in the field of spread spectrum radar poly-phase codes [9]. The four competitor algorithms have been applied on this problem. We omit the detailed description of the associated fitness function in order to save space.

Table 1. Benchmark Functions Used

Function	Mathematical Representation
Rosenbrock	$f_1(x) = \sum_{i=1}^{n-1} [100\,(x_{i+1} - x_i^{\,2})^2 + (x_i - 1)^2]$
Rastrigin	$f_2(x) = \sum_{i=1}^{n} [\,x_i^{\,2} - 10\,\cos(2\pi x_i) + 10\,]$
Griewank	$f_3(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^{\,2} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$
Ackley	$f_4(x) = -20\exp(-0.2\sqrt{(\frac{1}{n}\sum_{i=1}^{n} x_i^2)} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i) + 20 + e$
Shekel's Foxholes	$f_5(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}]^{-1}$

3.2 Simulation Strategy

The proposed BSO algorithm has been compared with the classical PSO, original BFOA and a recently developed variant of PSO known as MPSO-TVAC [13]. In the later version velocity of a randomly selected particle is perturbed by a random mutation step size if the global best-sofar solution does not improve for a predetermined number of generations. Following [13] we keep the mutation step size proportional to the maximum allowable velocity. For all the competitive algorithms we use same population size which amounts to 40 particles or bacteria in corresponding algorithm .To make the comparison fair population all the competitor algorithms (all problems tested) were initialized using the same random seed.

We choose the number of fitness function evaluations (FEs) as a measure of the computational time instead of 'iterations' or 'generations'. Twenty-five independent runs of the four competitor algorithms were carried out on each problem and the average of the best -of- run solutions and standard deviations were noted. Each run was continued for different maximum number of FEs depending on the complexity of the problem. The spread spectrum radar poly-phase code design problem was tested varying n from 2 to 20. We, however, report result of just two of the most difficult problem instances (for dimensions 19 and 20) owing to the space limitations. Standard set of parameters was used for the PSO algorithm and the original BFOA. In case of the BSO algorithm we have chosen the best-suited set of parameters after a series of hand tuning experiments. We take N_{re}=4, N_c=50, ω=0.8, C_1=C_2=1.494. The same set of parameters was used for all algorithms. For BFOA and MPSO-TVAC, we have employed the standard set of parameter values as recommended in [1] and [13] respectively.

4 Results

Table 2 compares the algorithms on quality of the optimum solution over five benchmarks. The mean and the standard deviation (within parenthesis) of the

Table 2. Mean and Standard Deviation over five benchmarks

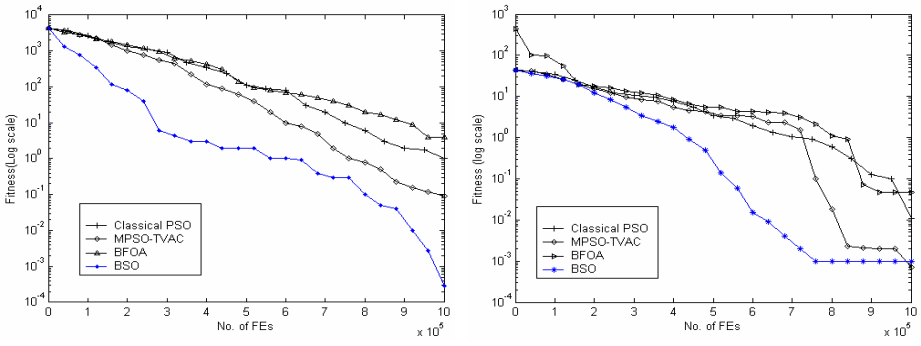
Fun	Dim	Max^m FE	Mean Best Value (Standard Deviation)			
			BFOA	Classical PSO	MPSO-TVAC	BSO
f_1	15	50,000	26.705 (2.162)	14.225 (3.573)	4.217 (1.332)	**0.483 (0.074)**
	30	1×10^5	58.216 (14.32)	46.139 (9.649)	22.432 (7.178)	**15.471 (2.655)**
	45	5×10^5	96.873 (26.136)	83.630 (14.536)	43.258 (16.944)	**27.986 (4.338)**
	60	1×10^6	154.705 (40.162)	122.239 (67.728)	97.537 (24.379)	**52.263 (8.341)**
f_2	15	50,000	6.9285 (2.092)	3.3484 (0.297)	1.1545 (0.321)	**0.082 (0.00928)**
	30	1×10^5	17.0388 (4.821)	12.7374 (0.781)	**9.8824 (0.931)**	10.2266 (0.238)
	45	5×10^5	30.9925 (7.829)	24.8286 (1.818)	17.0656 (1.352)	**13.5034 (3.923)**
	60	1×10^6	45.8234 (9.621)	36.3343 (6.291)	22.253 (4.889)	**18.3621 (5.773)**
f_3	15	50,000	0.2812 (0.0216)	**0.0361 (0.00524)**	0.1613 (0.097)	0.0541 (0.0287)
	30	1×10^5	0.3729 (0.046)	0.1348 (0.107)	0.2583 (0.1232)	**0.0792 (0.0113)**
	45	5×10^5	0.6351 (0.052)	0.1969 (0.116)	0.5678 (0.236)	**0.1352 (0.0135)**
	60	1×10^6	0.8324 (0.076)	0.7584 (0.342)	0.6113 (0.097)	**0.2547 (0.0287)**
f_4	15	50,000	0.9332 (0.0287)	0.5821 (0.0542)	0.1696 (0.0026)	**0.0825 (0.0007)**
	30	1×10^5	4.3243 (1.883)	0.8578 (0.042)	0.7372 (0.0415)	**0.5921 (0.036)**
	45	5×10^5	12.4564 (3.434)	1.8981 (0.195)	**0.8922 (0.1453)**	0.9383 (0.1327)
	60	1×10^6	8.3247 (1.613)	2.4062 (0.451)	2.1692 (0.418)	**1. 8766 (0.536)**
f_6	2	50,000	0.999868 (0.00217)	0.999832 (0.00167)	0.999805 (0.00485)	**0.999800 (0.0000)**

best-of-run values of 25 independent runs for each of the four algorithms were presented. Each algorithm is predetermined maximum number of FEs. The best solution in each case has been marked in bold. Table 3 presents results of the unpaired *t*-tests between BSO and best of the three competitive algorithms in each case (standard error of difference of the two means, 95% confidence interval of this difference, the *t* and the two tailed P value). In table 3 for all cases the sample size =25 and degrees of freedom = 48. It is interesting to note from table 2 and 3 that for most of the cases the BSO algorithm meets or beats its nearest competitor in a statistically meaningful way. Table 2 shows that in three cases (f_2 (30), f_3 (15), f_4 (45)) the mean of the BSO algorithm is greater than that of the classical PSO or MPSO-TVAC. But table 3 reveals that in two cases i.e. f_2 (30) and f_4 (45) this difference is not statistically significant. Also one may perceive that incorporating the g_best PSO

operator besides the computational chemotaxis has significantly improved the performance of BSO as compared to the original BFOA algorithm. Tables 4 and 5 present the corresponding results for the radar poly-phase code design problem for n=19 and n=20. In figure 1 we have graphically presented the rate of convergence of the competitor algorithms for all the functions in 30 dimensions. The graphs have been drawn for the median of the run for all cases. Figure 2 shows the scalability of the four methods on two tests functions-how the average time of convergence varies with the dimensionality of the search space. We omit rest of test functions for the sake of space economy. The graphs suggest that the effects of the *curse of dimensionality* are comparable on BSO and MPSO-TVAC and at least far better than classical PSO and BFOA.

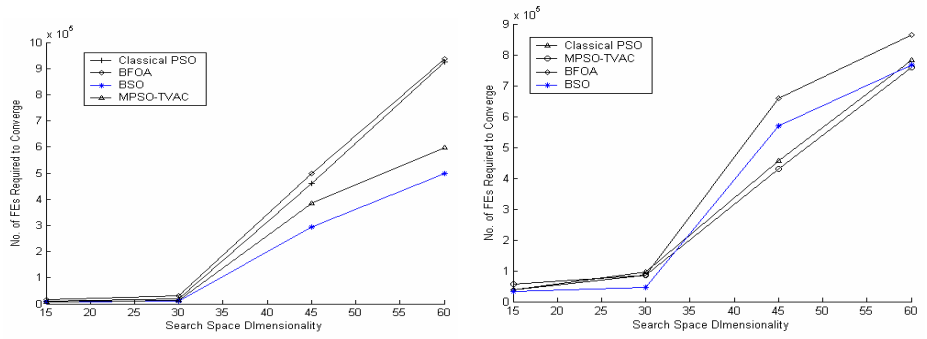
Table 3. Results of unpaired t-tests on the data of Table 3

Fn, Dim	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
f_1, 15	0.267	13.9949	-4.27046 to -3.19754	< 0.0001	**Extremely significant**
f_1, 30	1.531	4.5477	-10.03859 to -3.88341	< 0.0001	**Extremely significant**
f_1, 45	3.498	4.3658	-22.30540 to -8.23860	< 0.0001	**Extremely significant**
f_1, 60	5.153	8.7855	-55.63537 to -34.91263	< 0.0001	**Extremely significant**
f_2, 15	0.064	16.6908	-1.2011367 to -0.9428633	<0.0001	**Extremely Significant**
f_2, 30	0.192	1.7899	-0.04242 to 0.73042	0.0798	Not quite Significant
f_2, 45	1.513	14.4685	-24.9330 to -18.8487	< 0.0001	**Extremely significant**
f_2, 60	1.513	2.5724	-6.932087 to -0.849713	0.0132	Significant
f_3, 15	0.006	3.0849	0.0062682 to 0.0297318	0.0034	Very Significant
f_3, 30	0.022	2.5838	0.012333 to 0.098867	0.0129	Significant
f_3, 45	0.023	2.6417	-0.108662 to -0.014738	0.0111	Significant
f_3, 60	0.020	17.6261	-0.3972779 to -0.3159221	<0.0001	**Extremely Significant**
f_4, 15	0.001	161.740	-0.0881827 to -0.086017	<0.0001	**Extremely Significant**
f_4, 30	0.011	13.2057	-0.1671923 to -0.1230077	<0.0001	**Extremely Significant**
f_4, 45	0.030	0.8735	-0.033979 to 0.086179	0.3867	Not Significant
f_4, 60	0.136	2.1524	-0.565934 to -0.019266	0.0364	Significant
f_5,2	0.001	0.0052	-0.0019553 to 0.0019453	0.9959	Not Significant



(a) Rastrigin (f_2) (b) Ackley (f_4)

Fig. 1. Progress towards the optima for two benchmark functions



(a) Rastrigin (f_2) (b) Ackley (f_4)

Fig. 2. Variation of computational cost with search space dimensionality

Table 4. Average and the standard deviation of the best-of-run solution for 25 runs for spread spectrum radar poly-phase code design problem (number of dimensions n = 19 and n = 20). For all cases each algorithm was run for 50,000 FEs.

N	Mean best-of-run solution (Std Dev)			
	BFOA	Classical PSO	MPSO-TVAC	BSO
19	0.7974 (0.0323)	0.7524 (0.00493)	0.7832 (0.00038)	**0.7519 (0.0392)**
20	0.8577 (0.0283)	0.8693 (0.0048)	0.8398 (0.0482)	**0.8134 (0.0482)**

Table 5. Results of unpaired t-tests on the data of Table 4

n	Std. Err	*t*	95% Conf. Intvl	Two-tailed *P*	Significance
19	0.002	2.5024	-0.00734 to -0.00081	0.0152	Significant
20	0.010	3.5880	-0.05792 to -0.01644	0.0007	**Extremely significant**

5 Conclusions

The paper has presented an improved variant of the BFOA algorithm by combining the PSO based mutation operator with bacterial chemotaxis. The present scheme attempts to make a judicious use of exploration and exploitation abilities of the search space and therefore likely to avoid false and premature convergence in many cases. The overall performance of the proposed algorithm is definitely better than a standalone BFOA at least on the numerical benchmarks tested. The performance also appears to be at least comparable with PSO and its variants. The future research effort should focus on reducing the number of user-defined parameters for BFOA and its variants. Also an empirical study on the effects of these parameters on the convergence behavior of the hybrid algorithms may be worthy to undertake.

References

1. Passino, K.M.: Biomimicry of Bacterial Foraging for Distributed Optimization and Control, IEEE Control Systems Magazine, 52-67, (2002).
2. Mishra, S.: A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. IEEE Trans. on Evolutionary Computation, vol. 9(1): 61-73, (2005).
3. Tripathy, M., Mishra, S., Lai, L.L. and Zhang, Q.P.: Transmission Loss Reduction Based on FACTS and Bacteria Foraging Algorithm. PPSN, 222-231, (2006).
4. Kim, D.H., Cho, C. H.: Bacterial Foraging Based Neural Network Fuzzy Learning. IICAI 2005, 2030-2036.
5. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975).
6. Kennedy, J, Eberhart, R.: Particle swarm optimization, In Proceedings of IEEE International Conference on Neural Networks, (1995) 1942-1948.
7. Storn, R., Price, K.: Differential evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization, 11(4) 341–359, (1997).
8. Kim, D.H., Abraham, A., Cho, J.H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization, Information Sciences, Vol. 177 (18), 3918-3937, (2007).
9. Mladenovic, P., Kovacevic-Vuijcic, C.: Solving spread-spectrum radar polyphase code design problem by tabu search and variable neighborhood search, European Journal of Operational Research, 153(2003) 389-399.
10. Stephens, D.W., Krebs, J.R., Foraging Theory, Princeton University Press, Princeton, New Jersey, (1986).
11. Yao, X., Liu, Y., Lin, G. Evolutionary programming made faster, IEEE Transactions on Evolutionary Computation, vol 3, No 2, 82-102, (1999).
12. Angeline, P. J.:Evolutionary optimization versus particle swarm optimization: Philosophy and the performance difference, Lecture Notes in Computer Science (vol. 1447), Proceedings of 7th International Conference on. Evolutionary Programming – Evolutionary Programming VII (1998) 84-89.
13. Ratnaweera, A., Halgamuge, K.S.: Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, In IEEE Transactions on Evolutionary Computation 8(3): 240-254, (2004).