Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis

Sambarta Dasgupta, Swagatam Das, Ajith Abraham, *Senior Member, IEEE*, and Arijit Biswas

***Abstract*—In his seminal paper published in 2002, Passino pointed out how individual and groups of bacteria forage for nutrients and how to model it as a distributed optimization process, which he called the bacterial foraging optimization algorithm (BFOA). One of the major driving forces of BFOA is the chemotactic movement of a virtual bacterium that models a trial solution of the optimization problem. This paper presents a mathematical analysis of the chemotactic step in BFOA from the viewpoint of the classical gradient descent search. The analysis points out that the chemotaxis employed by classical BFOA usually results in sustained oscillation, especially on flat fitness landscapes, when a bacterium cell is close to the optima. To accelerate the convergence speed of the group of bacteria near the global optima, two simple schemes for adapting the chemotactic step height have been proposed. Computer simulations over several numerical benchmarks indicate that BFOA with the adaptive chemotactic operators shows better convergence behavior, as compared to the classical BFOA. The paper finally investigates an interesting application of the proposed adaptive variants of BFOA to the frequency-modulated sound wave synthesis problem, appearing in the field of communication engineering.**

***Index Terms*—Bacterial foraging, biological systems, computational chemotaxis, global optimization, gradient descent search, swarm intelligence.**

I. Introduction

To TACKLE complex search problems of the real world, scientists have been drawing inspiration from nature and natural creatures for years. Optimization is at the heart of many natural processes like Darwinian evolution, group behavior of social insects, and the foraging strategy of other microbial creatures. Natural selection tends to eliminate species with poor foraging strategies and favor the propagation of genes of species with successful foraging behavior since they are more likely to enjoy reproductive success.

Since a foraging organism or animal takes necessary action to maximize the energy intake per unit time spent for foraging, considering all the constraints presented by its own physiology such as sensing and cognitive capabilities, environment (e.g., density of prey, risks from predators, physical characteristics of the search space), the natural foraging strategy can lead to optimization and essentially this idea can be applied to solve real-world optimization problems [1]. Based on this concept, Passino proposed an optimization technique known as the bacterial foraging optimization algorithm (BFOA) [1], [2]. To date, BFOA has successfully been applied to real-world problems such as optimal controller design [1], [3], harmonic estimation [4], transmission loss reduction [5], active power filter synthesis [6], and learning of artificial neural networks [7].

One major step in BFOA is the simulated chemotactic movement. Chemotaxis is a foraging strategy that implements a type of local optimization, where the bacteria try to climb up the nutrient concentration to avoid noxious substance and search for ways out of neutral media. This step has much resemblance with a biased random walk model [8].

This paper provides a mathematical analysis of the simulated chemotaxis in the light of the classical gradient descent search [9], [10] algorithm. The analysis reveals that a chemotactic step height varying as the function of the current fitness value can lead to better convergence behavior as compared to a fixed step height. The adaptation schemes, proposed for automatically adjusting the step height, are simple and impose no additional computational burden on the BFOA in terms of excess number of function evaluations (FEs). At this point, we would like to mention that several researchers have investigated the adaptation of step size in both stochastic and deterministic gradient descent optimization algorithms [11]–[16] mostly in the context of training neural networks. However, unlike the works cited in [11]–[16], the adaptation schemes proposed in this paper for the chemotactic step height are not based on complex calculus techniques, does not require computationally intensive operations such as Hessian matrix evaluation [17], [18], and is general enough for widespread use with BFOA as a black-box optimization technique. They are solely based on the fitness information of each individual member of the bacterial population and accelerate the convergence of the bacterium to an optimum, avoiding any oscillatory behavior around the optimum.

The proposed adaptive BFOA (ABFOA) schemes have been compared with their classical counterpart, a very popular swarm-intelligence algorithm known as the particle swarm optimization (PSO) [19] and a standard real-coded genetic algorithm (GA) [20], [21] over a test suite of ten numerical benchmarks with respect to the following performance measures: solution quality, convergence speed, and the frequency of hitting the optimal solution. This paper also investigates an interesting application of the ABFOA schemes to the parameter estimation of frequency-modulated sound waves.

Manuscript received September 22, 2007; revised February 29, 2008 and May 18, 2008; accepted May 20, 2008. Current version published August 14, 2009.

S. Dasgupta, S. Das, and A. Biswas are with the Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India (e-mail: sambartadg@gmail.com; swagatamdas19@yahoo.co.in; arijitbiswas87@gmail.com).

A. Abraham is with the Center for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway (e-mail: ajith.abraham@ieee.org).

Digital Object Identifier: 10.1109/TEVC.2009.2021982

The rest of the paper is organized as follows. In Section II, we outline the classical BFOA in sufficient details. Section III reviews the state-of-the-art research works on BFOA. Section IV provides a mathematical analysis of the chemotactic movement of a simple 1-D bacterium and proposes the adaptive chemotactic operators for BFOA. Section V provides detailed comparison between the classical BFOA and its adaptive variants over a test suite of 10 well-known numerical benchmarks. Section VI describes an application of the adaptive BFOAs to the parameter estimation problem for frequency-modulated sound wave. Finally, conclusions are drawn in section VII. Analysis of the chemotaxis for a multibacterial system has been presented in Appendix I. It has also been shown that under certain conditions the mathematical model of the multibacterial system boils down to that of a single bacterium.

II. Classical BFOA Algorithm

The bacterial foraging system consists of four principal mechanisms, namely chemotaxis, swarming, reproduction, and elimination dispersal [1]. Below we briefly describe each of these processes and finally provide a pseudo-code of the complete algorithm.

A. Chemotaxis

This process simulates the movement of an *E.coli* cell through swimming and tumbling via flagella. Biologically, an *E.coli* bacterium can move in two different ways. It can swim for a period of time in the same direction, or it may tumble, and alternate between these two modes of operation for the entire lifetime. Suppose $\theta^i(j,k,l)$ represents ith bacterium at jth chemotactic, kth reproductive and lth elimination dispersal step. $C(i)$ is the size of the step taken in the random direction specified by the tumble (run length unit). Then in computational chemotaxis the movement of the bacterium may be represented by

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

where Δ indicates a vector in the random direction whose elements lie in [–1, 1].

B. Swarming

An interesting group behavior has been observed for several motile species of bacteria including *E.coli* and *S. typhimurium*, where intricate and stable spatiotemporal patterns (swarms) are formed in a semisolid nutrient medium [22], [23]. A group of *E.coli* cells arrange themselves in a traveling ring by moving up the nutrient gradient when placed amidst a semisolid matrix with a single nutrient chemoeffecter. The cells, when stimulated by a high level of *succinate*, release an attractant *aspertate*, which helps them to aggregate into groups and thus move as concentric patterns of swarms with high bacterial density. The cell-to-cell signaling in *E. coli* swarm may be represented by the following function:

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

where $J_{cc}(\theta, P(j,k,l))$ is the objective function value to be added to the actual objective function (to be minimized) to present a time-varying objective function, S is the total number of bacteria, p is the number of variables to be optimized that are present in each bacterium, and $\theta = [\theta_{1,}\theta_{2,\ldots,}\theta_p]^T$ is a point in the p-dimensional search domain. $d_{aatractant}, w_{attractant}, h_{repellant}, w_{repellant}$ are different coefficients that should be chosen properly.

C. Reproduction

The least healthy bacteria eventually die while each of the healthier bacteria (those yielding lower value of the objective function) asexually split into two bacteria, which are then placed in the same location. This keeps the swarm size constant.

D. Elimination and Dispersal

Gradual or sudden changes in the local environment where a bacterium population lives may occur due to various reasons: e.g., a significant local rise of temperature may kill a group of bacteria that are currently in a region with a high concentration of nutrient gradients. Events can take place in such a fashion that all the bacteria in a region are killed or a group is dispersed into a new location. To simulate this phenomenon in BFOA, some bacteria are liquidated at random with a very small probability while the new replacements are randomly initialized over the search space.

The detailed pseudo-code and a flowchart (Fig. 1) of the complete algorithm is given below:

The BFOA Algorithm

```
Step 1. Initialize parameters $p, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, C(i)(I = 1, 2, \ldots, S), \theta^i$
        where
        $p$: dimension of the search space,
        $S$: total number of bacteria in the population,
        $Nc$: number of chemotactic steps,
        $N_s$: swimming length,
        $N_{re}$: the number of reproduction steps,
        $N_{ed}$: the number of elimination–dispersal events,
        $P_{ed}$: elimination-dispersal probability, and
        $C(i)$: the size of the step taken in the random direction specified by the tumble.
Step 2. Elimination–dispersal loop: $l = l + 1$.
Step 3. Reproduction loop: $k = k + 1$.
Step 4. Chemotaxis loop: $j = j + 1$.
```

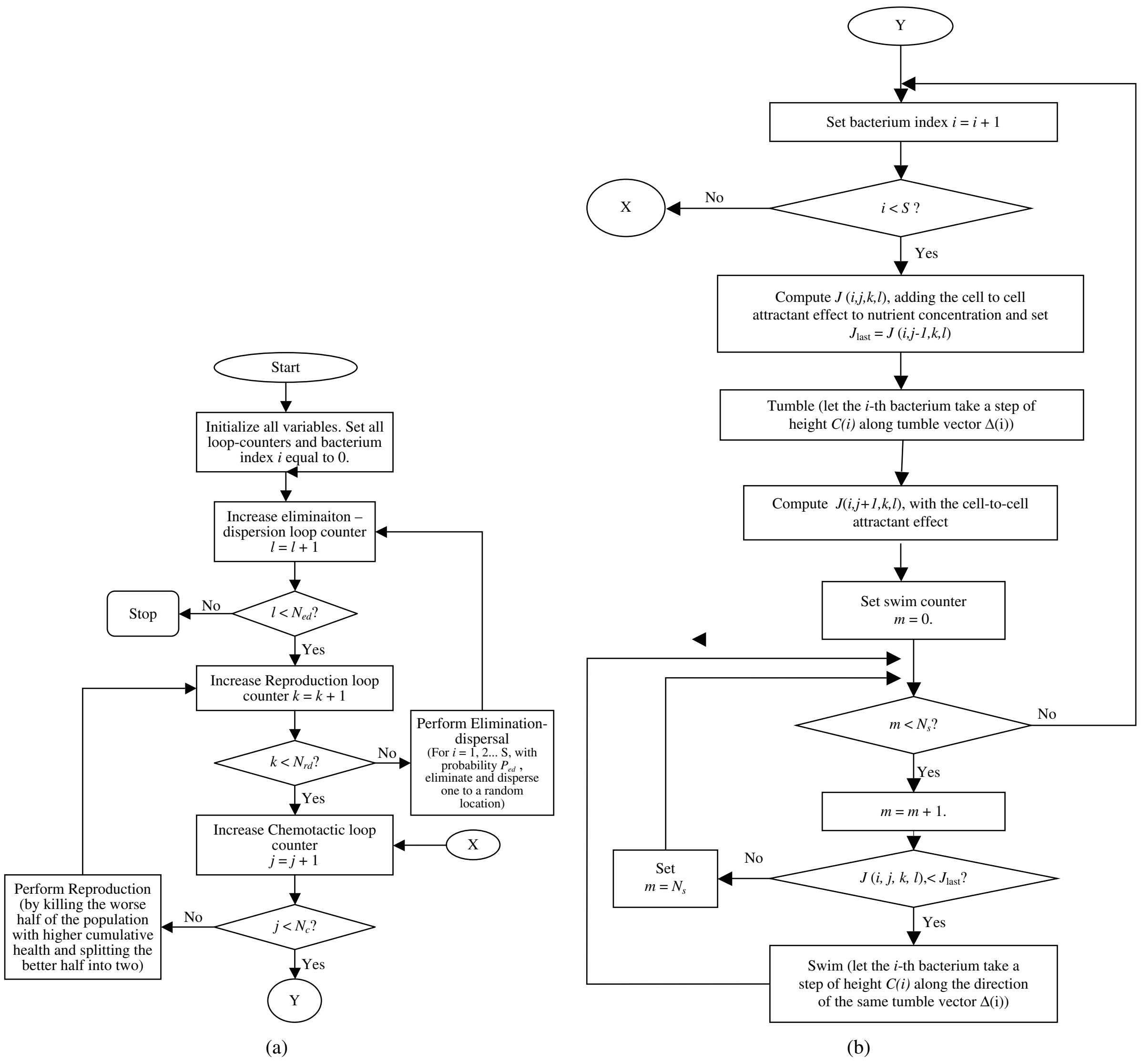

Start
Initialize all variables. Set all loop-counters and bacterium index i equal to 0.
Increase eliminaiton – dispersion loop counter l = l + 1
l < N_ed?
No
Stop
Yes
Increase Reproduction loop counter k = k + 1
k < N_rd?
No
Perform Elimination-dispersal (For i = 1, 2... S, with probability P_ed, eliminate and disperse one to a random location)
Yes
Increase Chemotactic loop counter j = j + 1
X
j < N_c?
No
Perform Reproduction (by killing the worse half of the population with higher cumulative health and splitting the better half into two)
Yes
Y
(a)
Y
Set bacterium index i = i + 1
i < S ?
No
X
Yes
Compute J (i,j,k,l), adding the cell to cell attractant effect to nutrient concentration and set J_last = J (i,j-1,k,l)
Tumble (let the i-th bacterium take a step of height C(i) along tumble vector Δ(i))
Compute J(i,j+1,k,l), with the cell-to-cell attractant effect
Set swim counter m = 0.
m < N_s?
No
Yes
m = m + 1.
J (i, j, k, l),< J_last?
No
Set m = N_s
Yes
Swim (let the i-th bacterium take a step of height C(i) along the direction of the same tumble vector Δ(i))
(b)

Fig. 1. Flowchart of the bacterial foraging algorithm.

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

(b) Compute fitness function $J\ (i, j, k, l)$.
Let, $J(i, j, k, l) = J(i, j, k, l) + J_{cc}(\theta^i(j, k, l), P(j, k, l))$ (i.e., add on the cell-to cell attractant–repellant profile to simulate the swarming behavior) where J_{cc} is defined in (2).

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

(d) 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].

(e) Move: Let

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

this results in a step of size $C(i)$ in the direction of the tumble for bacterium i.

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

(g) Swim.
  i) Let $m = 0$ (counter for swim length);
  ii) While $m < N_s$ (if have not climbed down too long);
  iii) Let $m = m + 1$;
  iv) If $J(i, j+1, k, l) < J_{last}$ (if doing better), let $J_{last} = J(i, j+1, k, l)$ and let

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

  And use this $\theta^i(j+1, j, k)$ to compute the new $J(i, j+1, k, l)$ as we did in (f);
  v) Else, let $m = N_s$. This is the end of the while statement.

(h) Go to next bacterium $(I + 1)$ if $i \neq S$ [i.e., go to (b) to process the next bacterium].

Step 5. If $j < N_c$, go to Step 4. In this case, continue chemotaxis since the life of the bacteria is not over.

Step 6. Reproduction:

(a) For the given k and l, and for each $i = 1, 2, \ldots, S$, let

$$J^i_{\text{health}} = \sum_{j=1}^{N_c+1} J(i, j, k, l) \tag{3}$$

be the health of the 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 parameters $C(i)$ in order of ascending cost J_{health} (higher cost means lower health).

(b) The S_r bacteria with the highest J_{health} values die and the remaining S_r bacteria with the best values split (this process is performed by placing the copies that are made at the same location as their parent).

Step 7. If $k < N_{re}$, go to Step 3. In this case, we have not reached the number of specified reproduction steps, so we start the next generation of the chemotactic loop.

Step 8. Elimination–dispersal: For $i = 1, 2 \ldots, S$ with probability P_{ed}, eliminate and disperse each bacterium (this keeps the number of bacteria in the population constant). To do this, if a bacterium is eliminated, simply disperse another one to a random location on the optimization domain. If $l < N_{ed}$, then go to Step 2; otherwise end.

III. Related Works on BFOA

Since its advent in 2002, BFOA has attracted researchers from diverse domains of knowledge. This has resulted in a few variants of the classical algorithm as well as many interesting applications of the same to the real-world optimization problems. In 2002, Liu and Passino [2] incorporated a new function $J_{ar}(\theta)$ in BFOA to represent the environment-dependent cell-to-cell signaling, such that

$$J_{ar}(\theta) = \exp(M - J(\theta)).J_{cc}(\theta) \tag{4}$$

where M is a tunable parameter and $J_{cc}(\theta)$ is given by (2). For swarming, they considered the minimization of $J(i, j, k, l) + J_{ar}(\theta^i)$.

Tang *et al.* [24] proposed a bacterial foraging behavior in varying environments. Their study focused on the use of an individual-based modeling (IbM) method to simulate the activities of bacteria and the evolution of bacterial colonies. They derived a bacterial chemotaxis algorithm in the same framework and showed through simulation studies that the proposed algorithm could reflect the bacterial behaviors and population evolution in varying environments. Li *et al.* proposed a modified bacterial foraging algorithm with varying population (BFAVP) [25] and applied the same to the optimal power flow (OPF) problems. Instead of simply describing chemotactic behavior into BFOA, as was done by Passino [1], BFAVP also incorporates the mechanisms of bacterial proliferation and quorum sensing, which allow a varying population in each generation of bacterial foraging process.

Tripathy and Mishra proposed an improved BFO algorithm for simultaneous optimization of the real power losses and voltage stability limit (VSL) of a mesh power network [26]. In their modified algorithm, first, instead of the average value, the minimum value of all the chemotactic cost functions is retained for deciding the bacterium's health. This speeds up the convergence, because in the average scheme described by Passino [1] it may not retain the fittest bacterium for the subsequent generation. Second, for swarming the distances of all the bacteria in a new chemotactic stage are evaluated from the globally optimal bacterium to these points and not the distances of each bacterium from the rest of the others, as suggested by Passino [1]. Simulation results indicated the superiority of the proposed approach over the classical BFOA for the multiobjective optimization problem involving the unified power flow controller (UPFC) location, its series-injected voltage, and the transformer tap positions as the variables. Mishra and Bhende used the modified BFOA to optimize the coefficients of proportional integral (PI) controllers for active power filters [6]. The proposed algorithm was found to outperform a conventional GA with respect to the convergence speed.

Mishra [4] proposed a Takagi–Sugeno-type fuzzy inference scheme for selecting the optimal chemotactic step size in BFOA. The resulting algorithm, referred to as "fuzzy bacterial foraging" (FBF), was shown to outperform both classical BFOA and a GA when applied to the harmonic estimation problem. However, the performance of the FBF crucially depends on the choice of the membership function and the fuzzy rule parameters [4], and there is no systematic method (other than trial and error) to determine these parameters for a given problem. Hence FBF, as presented in [4], may not be suitable for optimizing any benchmark function in general.

Hybridization of BFOA with other naturally inspired metaheuristics has remained an interesting problem for the researchers. In this context, Kim *et al.* proposed a hybrid approach involving GA and BFOA for function optimization [3]. The proposed algorithm outperformed both GA and BFOA over a few numerical benchmarks and a practical PID controller design problem. Biswas *et al.* proposed a synergism of BFOA with another very popular swarm intelligence algorithm well known as the particle swarm optimization (PSO). The new algorithm, named by the authors as "bacterial swarm optimization" (BSO) [27], was shown to perform in a statistically significantly better way as compared to both of its classical counterparts over several numerical benchmarks.

Ulagammai *et al.* applied BFOA to train a wavelet-based neural network (WNN) and used the same for identifying the inherent nonlinear characteristics of power system loads [28]. Munoz *et al.* [29] used BFOA for the dynamical resource allocation in a multiple input/output experimentation platform, which mimics a temperature grid plant and is composed of multiple sensors and actuators organized in zones. Acharya *et al.* proposed a BFOA-based independent component analysis (ICA) [30] that aims at finding a linear representation of non-Gaussian data so that the components are statistically independent or as independent as possible. The proposed scheme yielded better mean square error performance as compared

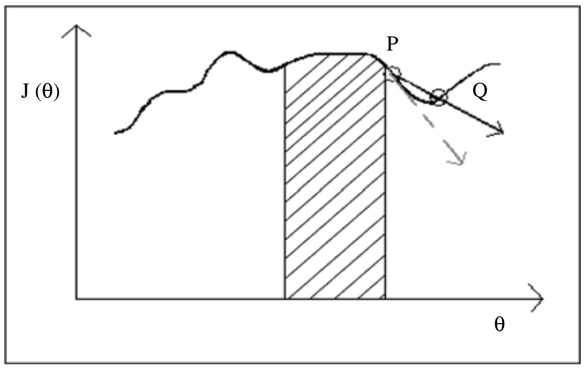


Fig. 2. Continuous 1-D fitness landscape for BFOA. The analysis presented here holds perfect for regions like the shaded one.

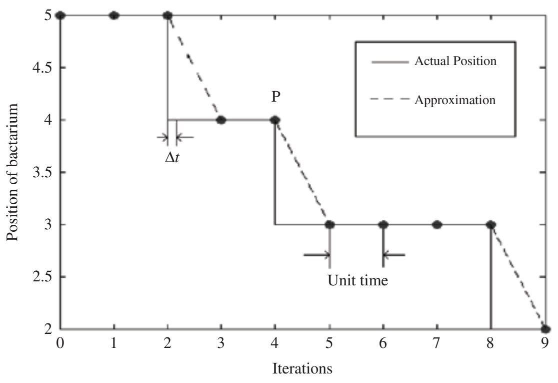


Fig. 3. Bacterium changing positions instantaneously and its approximated counterpart.

to a constrained genetic algorithm-based ICA (CGAICA). Chatterjee *et al.* [31] reported an interesting application of BFOA to improve the quality of solutions for the extended Kalman filters (EKFs), such that the EKFs can offer solutions to simultaneous localization and mapping (SLAM) problems for mobile robots and autonomous vehicles.

To the best of our knowledge, none of the existing works has, however, attempted to develop a full-fledged mathematical model of the bacterial foraging strategies for investigating important issues related to convergence, stability, and oscillations of the foraging dynamics near global optima.

IV. Simple Algebraic Analysis of the Computational Chemotaxis

Let us consider a single bacterium cell that undergoes chemotactic steps according to (1) over a 1-D objective function. The bacterium lives in continuous time and at the tth instant its position is given by, which is a real number. Let the objective function to be minimized be $J(x) = x^2$. Below we list a few assumptions that were considered for the sake of gaining mathematical insight into the process.

1) The objective function $J(x)$ is continuous and differentiable at all points in the search space. (Note that BFOA is a derivative-free optimization technique and can in general handle discontinuous and nondifferentiable functions also. We, however, make the above assumption in order to simplify the mathematical manipulations.)
2) The chemotactic step size C is not very large (Passino himself took $C = 0.1$).
3) The analysis applies to the regions of the fitness landscape where gradients of the function are small, i.e., near the optima.

According to assumption 3), the analysis will be restricted to within regions like the shaded one in Fig. 2. In Fig. 2, the dashed line arrow represents velocity of the bacterium and the blue arrow shows the gradient vector. We note that the velocity vector does not necessarily coincide with the gradient vector. Initially, the bacterium was at point P and it moves to point Q. Here, the vector **PQ** shows the direction of velocity of the bacterium.

A. Analytical Treatment

Let the position of an individual bacterium at time t be $\theta(t)$ and value of objective function $J(\theta)$. As $\theta(t)$ is a function of time, the value of the objective function associated with this bacterium $J(\theta(t))$ is also changing with time. The bacterium may change its position continuously with time. Computer simulation of the algorithm, however, proceeds through discrete iterations (chemotactic steps). Certain amount of processor time is elapsed between two successive iterations. Thus, in the virtual world of simulations, a bacterium can change its position only at certain discrete time instants. This change of position is ideally instantaneous. In between two successive iterations the bacterium remains practically immobile. Without losing generality, the time between two successive iterations may be defined as unit time for the following derivation. The situation is depicted in Fig. 3, where the bacterium changes its position instantly at certain discrete time points. But we have assumed that the bacterium lives in continuous time, where it is not possible to have instantaneous position change. Hence, we assume that within two successive iterations the position shifts continuously and linearly [e.g., for time intervals (2–3), (4–5), (8–9)]. In practice, time between two successive iterations, i.e., computational time of iteration, is very small. This ensures that the linear approximation is fairly good.

Let us assume that, after an infinitesimal time intervalΔt, its position changes by an amount $\Delta\theta$ from $\theta(t)$ when the objective function value becomes smaller for changed position. Fig. 4 reveals the nature of the time rate of change of position of the bacterium. We can now define the velocity of the bacterium as

$$V_b = \underset{\Delta t \to 0}{Lim} \frac{\Delta\theta}{\Delta t}. \tag{5}$$

Naturally, here we assume the time to be unidirectional (i.e., $\Delta t > 0$).

From Fig. 4 it can be observered that the velocity of bacterium is a train of pulses occuring at certain points of time. As the pulse width is diminished its height is increased. Ideally, position changes instanteneously making height of the pulse tending toward infinity. Regardless of the pulse width, area under the rectangle is always equal to step height C (0.1

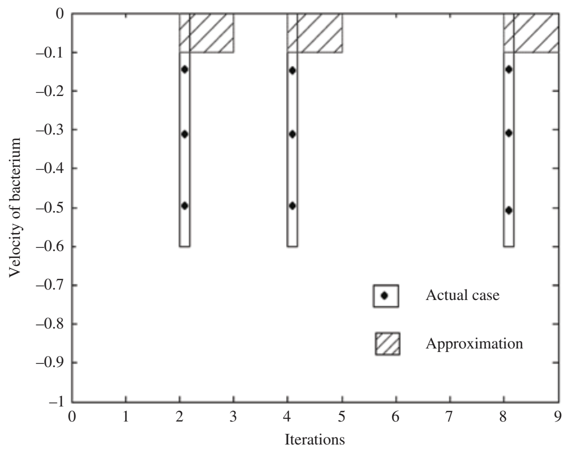


Fig. 4. Variation of velocity with successive iterations.

in this case). As we assume velocity to be constant over the complete interval (i.e., the position changes uniformly), its magnitude becomes C. In Figs. 3 and 4, iterations signify the successive chemotactic steps taken by the bacterium. Time between two consecutive iterations is elapsed in necessary computation associated with one step.

Now, according to BFOA, the bacterium changes its position only if the modified objective function value is less than the previous one i.e., $J(\theta) > J(\theta + \Delta\theta)$ i.e., $J(\theta) - J(\theta + \Delta\theta)$ is positive. This ensures that the bacterium always moves in the direction of decreasing objective function value. In this analysis we denote the unit vector along the direction of tumble by Δ, i.e., Δ here is analogous to the unit vector $\Delta(i)/\sqrt{\Delta^T(i).\Delta(i)}$ in (1) as used by Passino for a multidimensional fitness landscape. Note that for a 1-D problem, Δ is of unit magnitude and hence can assume only two values 1 or -1 with equal probabilities. Thus its value would remain unchanged after dividing it by square of its amplitude [as done in Step 4(e) of the classical BFO algorithm]. The bacterium moves by an amount of $C.\Delta$ if objective function value is reduced for new location. Otherwise, its position will not change at all. Assuming uniform rate of position change, if the bacterium moves $C.\Delta$ in unit time, its position is changed by $(C.\Delta)(\Delta t)$ in Δt seconds. It decides to move in the direction in which concentration of nutrient increases or, in other words, the objective function decreases i.e., $J(\theta) - J(\theta + \Delta\theta) > 0$. Otherwise, it remains immobile. We have assumed that Δt is an infinitesimally small positive quantity, and thus sign of the quantity $J(\theta) - J(\theta + \Delta\theta)$ remains unchanged if Δt divides it. So, the bacterium will change its position if and only if $(J(\theta) - J(\theta + \Delta\theta))/\Delta t$ is positive. This crucial decision-making (i.e., whether to take a step or not) activity of the bacterium can be modeled by a unit step function (also known as Heaviside step function [32]) defined as

$$\begin{aligned} u(x) &= 1, \quad \text{if } x > 0 \\ &= 0, \quad \text{otherwise} \end{aligned} \tag{6}$$

and thus,

$$\Delta\theta = u\left(\frac{J(\theta) - J(\theta + \Delta\theta)}{\Delta t}\right).(C.\Delta)(\Delta t). \tag{7a}$$

Evidently, $\Delta\theta = 0$ if $J(\theta) \le J(\theta + \Delta\theta)$ and $\Delta\theta = (C.\Delta).\Delta t$ if $J(\theta) > J(\theta + \Delta\theta)$.

Dividing both sides of (7a) by Δt we get

$$\frac{\Delta\theta}{\Delta t} = u\left[\frac{J(\theta) - J(\theta + \Delta\theta)}{\Delta t}\right].C.\Delta$$

$$\Rightarrow \frac{\Delta\theta}{\Delta t} = u\left[-\frac{\{J(\theta + \Delta\theta) - J(\theta)\}}{\Delta t}\right].C.\Delta. \tag{7b}$$

From (5) we have

$$V_b = \underset{\Delta t\to 0}{Lim}\frac{\Delta\theta}{\Delta t} = \underset{\Delta t\to 0}{Lim}\left[u\left\{-\frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta t}\right\}.C.\Delta\right]$$

$$\Rightarrow V_b = \underset{\Delta t\to 0}{Lim}\left[u\left\{-\frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta\theta}\frac{\Delta\theta}{\Delta t}\right\}.C.\Delta\right].$$

As $\Delta t \to 0$ makes $\Delta\theta \to 0$, we may write

$$V_b = u\left\{-\left(\underset{\Delta\theta\to 0}{Lim}\frac{J(\theta + \Delta\theta) - J(\theta)}{\Delta\theta}\right)\left(\underset{\Delta t\to 0}{Lim}\frac{\Delta\theta}{\Delta t}\right)\right\}.C.\Delta.$$

Again, $J(x)$ is assumed to be continuous and differentiable. $\underset{\Delta\theta\to 0}{Lim}(J(\theta + \Delta\theta) - J(\theta))/\Delta\theta$ is the value of the gradient at that point and may be denoted by $(dJ(\theta)/d\theta)$ or G. Therefore we have

$$V_b = u(-GV_b).C.\Delta \tag{8}$$

where $G = (dJ(\theta)/d\theta) =$ gradient of the objective function at θ. In (8), the argument of the unit step function is $-GV_b$. The value of the unit step function is 1 if G and V_b are of different sign, and in this case the velocity is $C\Delta$. Otherwise, it is 0, making the bacterium motionless. So (8) suggests that the bacterium will move in the direction of negative gradient. Since the unit step function $u(x)$ has a jump discontinuity at $x = 0$, to simplify the analysis further, we replace $u(x)$ with the continuous logistic function $\phi(x)$, where $\phi(x) = 1/(1 + e^{-kx})$. We note that

$$u(x) = \underset{k\to\infty}{Lt}\,\phi(x) = \underset{k\to\infty}{Lt}\frac{1}{1 + e^{-kx}}. \tag{9}$$

Fig. 5 illustrates how the logistic function may be used to approximate the unit step function used for decision making in chemotaxis. For analysis purpose, k cannot be infinity. We restrict ourselves to moderately large values of k (say, $k = 10$) for which $\phi(x)$ fairly approximates $u(x)$. Section IV-C describes the error limit introduced by this assumption. Thus, for moderately high values of k, $\phi(x)$ fairly approximates $u(x)$. Hence from (8)

$$V_b = \frac{C.\Delta}{1 + e^{kGV_b}}. \tag{10}$$

According to assumptions 1) and 3), if C and G are very small and $k \sim 10$, then also we may have $|kGV_b| << 1$. In that case we neglect the higher order terms in the expansion of e^{kgv_b} and have $e^{kgv_b} \approx 1 + kGV_b$. Substituting it in (10) we obtain

$$V_b = \frac{C.\Delta}{2 + kGV_b}$$

$$\Rightarrow V_b = \frac{C.\Delta}{2}\frac{1}{1 + \frac{kGV_b}{2}}$$

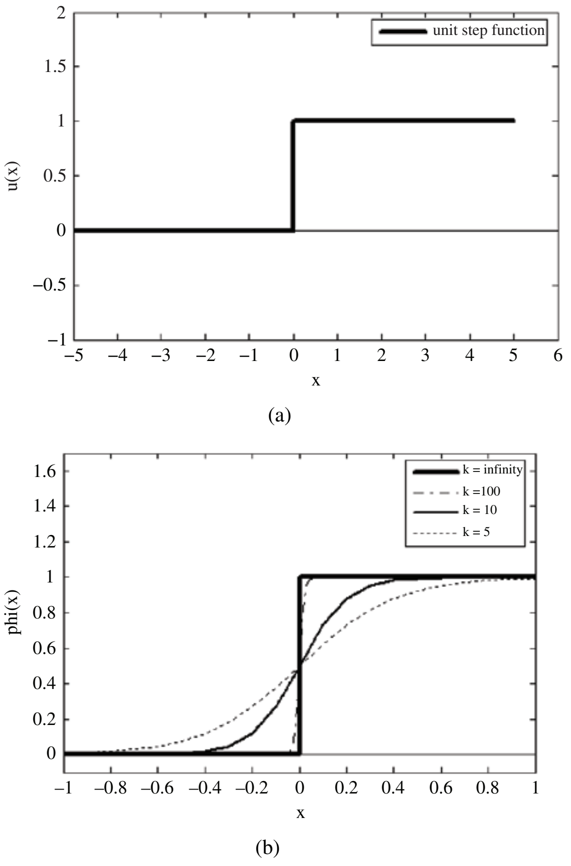


Fig. 5. (a) Unit step and (b) logistic functions.

$$\Rightarrow V_b = \frac{C.\Delta}{2}\left(1 - \frac{kGV_b}{2}\right)$$

$[\because |\frac{kGV_b}{2}| << 1$, neglecting higher terms, $(1 + \frac{kGV_b}{2})^{-1} \approx (1 - \frac{kGV_b}{2})]$.

After some manipulation we have

$$V_b = \frac{2C.\Delta}{4 + kGC.\Delta}.$$
$$\Rightarrow V_b = \frac{C.\Delta}{2}\frac{1}{1 + \frac{kCG\Delta}{4}}$$
$$\Rightarrow V_b = \frac{C.\Delta}{2}\left(1 - \frac{kGC\Delta}{4}\right) \tag{11}$$

$(\because |\frac{kGC\Delta}{4}| = |\frac{kGC}{4}| << 1$, as $|\Delta| = 1$ and neglecting the higher order terms.)

$$\Rightarrow V_b = \frac{C.\Delta}{2} - \frac{kGC^2\Delta^2}{8}$$
$$\Rightarrow V_b = -\frac{kC^2}{8}G + \frac{C.\Delta}{2}[\because \Delta^2 = 1]. \tag{12}$$

Equation (12) is applicable to a single bacterium system and it does not take into account the cell-to-cell signaling effect. A more complex analysis for the two-bacterium system involving the swarming effect has been included at the Appendix. It indicates that a complex perturbation term is added to the dynamics of each bacterium due to the effect of the neighboring bacteria cells. However, the term becomes negligibly small for small enough values of $C(\sim 0.1)$ and the dynamics under these circumstances get practically reduced to that described in (12). In what follows, we shall continue the analysis for single bacterium system for better understanding of the chemotactic dynamics.

B. Experimental Verification of Characteristic Equation of Chemotaxis

The characteristic equation of chemotaxis (12) represents the dynamics of bacterium taking chemotactic steps. In order to verify how reliably the equation represents the motion of the virtual bacterium, we compare results obtained from (12) with that of BFOA. First, the equation is expressed in iterative form, which is

$$V_b(n) = \theta(n) - \theta(n-1) = -\frac{kC^2}{8}G(n-1) + \frac{C.\Delta(n)}{2}$$
$$\Rightarrow \theta(n) = \theta(n-1) - \frac{kC^2}{8}G(n-1) + \frac{C.\Delta(n)}{2} \tag{13}$$

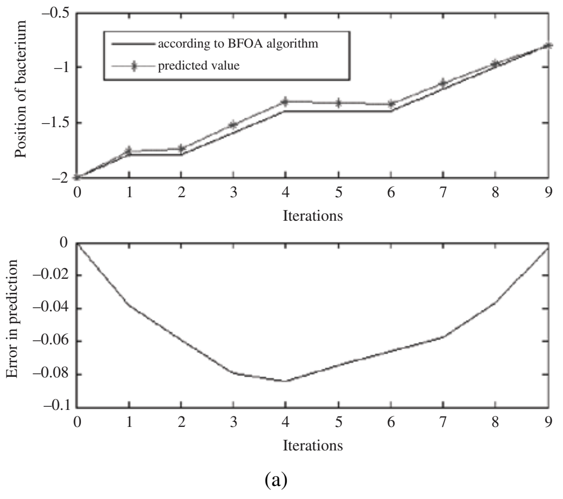
where n is the iteration index. The tumble vector is also a function of iteration count (i.e., chemotactic step number) i.e., it is generated repeatedly for successive iterations. We have taken $J(\theta) = \theta^2$ as the objective function for this experimentation. Bacterium was initialized at -2, i.e., $\theta(0) = -2$, and C is taken as 0.2. The gradient of $J(\theta)$ is 2θ. Therefore $G(n-1)$ may be replaced by $2\theta(n-1)$. Finally, for this specific case we get

$$\theta(n) = \left(1 - \frac{kC^2}{4}\right)\theta(n-1) + \frac{C.\Delta(n)}{2}. \tag{14}$$

We compute values of $\theta(n)$ for successive iterations according to above iterative relation. Also, values of positions are noted following guidelines of BFOA. The current position is changed by $C.\Delta$ if objective function value decreases for new position. The results are presented in Fig. 6. Fig. 6(a) shows the position in successive iterations according to BFOA and as obtained from (14). Here also we have assumed that the position of bacterium changes linearly between two consecutive iterations. Mismatch between actual and predicted values is also shown. In Fig. 6(b), the actual and predicted values of velocity are shown. Velocity is assumed to be constant between two successive iterations. According to BFOA, the magnitude of velocity is either C(0.2 in this case) or 0. The difference between the actual and predicted velocity is shown as the error. The time lapsed between two consequent iterations is spent for computation and is termed as “unit time.” This may be perceived as the time required by a bacterium to measure the nutrient content of a new point on the fitness landscape. Actually, it is the time taken by the processor to perform numerical computations.

C. Estimation of Error and Limitations of the Analysis

Due to the approximation of the unit step function, a small error has been introduced in the analysis. Again we have simplified the function for some special cases [assumptions 2) and 3)]. Here, magnitude of maximum possible value of error in estimation of V_b equals $|(C.\Delta)/2| = C/2. \because |\Delta| = 1$. According to (9) we may replace $u(x)$ approximately by the



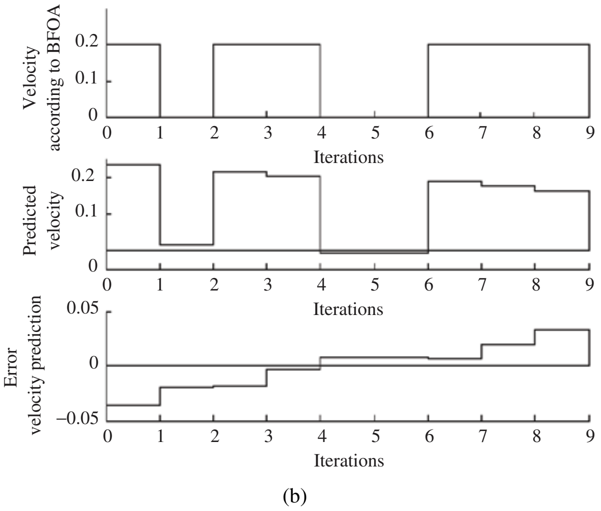


Fig. 6. Comparison between actual and predicted motional state of the bacterium. (a) Actual predicted positions of bacterium and error in estimation over successive iterations and (b) similar plots for velocity of the bacterium.

logistic function $\varphi(x) = 1/(1 + e^{-kx})$ for moderately high values of k. If x is small, we may again approximate the logistic function with the following equation of a straight line as:

$$\varphi(x) = \frac{k}{4}x + \frac{1}{2}. \tag{15}$$

These simplifications have already been undertaken in (10)–(12) of Section IV-A, where $x = GV_b$. The straight line, which approximates the logistic function as shown in Fig. 7, intersects the graph of the logistic function at two points A and B. When $|x| >$ OA or $|x| >$ OC, the error in the decision-making term of our analysis gradually increases. So we must restrict the analysis to within the region AC, i.e., magnitude of GV_b has certain limits. As shown in Fig. 7, x must lie between A and C, i.e., $\varphi(x)$ should be restricted within the range [0, 1], or otherwise considerable error creeps

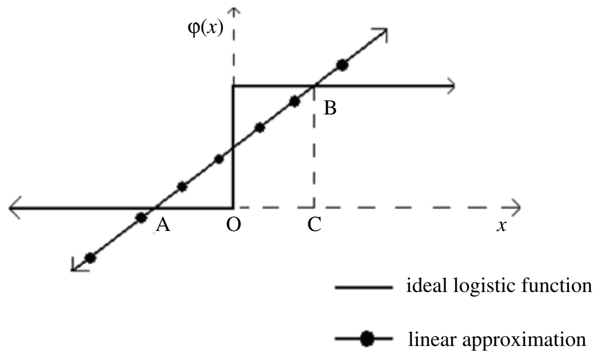


Fig. 7. Region of error due to approximation of the unit step with the logistic functions.

into the analysis. After imposing these constraints on (15), we get

$$x\frac{k}{4} + \frac{1}{2} \le 1 \quad \text{and} \quad x\frac{k}{4} + \frac{1}{2} \ge 0. \tag{16}$$

After solving above couple of inequalities, we finally obtain

$$|x| \le \frac{2}{k}. \tag{17}$$

Substituting x by GV_b in (17), we get

$$|V_b| \le \frac{2}{k|G|}. \tag{18}$$

From (12) and the above inequality, we get

$$\frac{2}{k|G|} \ge |V_b| = \left|\frac{C\Delta}{2} - \frac{kC^2}{8}G\right|. \tag{19}$$

We know any two numbers a and b, $|a - b| \ge |a| \sim |b|$. Hence

$$\left|\frac{C\Delta}{2} - \frac{kC^2}{8}G\right| \ge \left|\frac{C\Delta}{2}\right| - \left|\frac{kC^2}{8}G\right| = \frac{C}{2} - \frac{kC^2}{8}|G|.$$

[$C > 0, k > 0$ and Δ can assume values 1 and -1 randomly, giving $|\Delta| = 1$].

Incorporating inequality (19) in the above, we get

$$\frac{2}{k|G|} \ge \left|\frac{C\Delta}{2} - \frac{kC^2}{8}G\right| \ge \frac{C}{2} - \frac{kC^2}{8}|G|. \tag{20}$$

Again, as, $|a - b| \ge |b| - |a|$

$$\left|\frac{C\Delta}{2} - \frac{kC^2}{8}G\right| \ge \left|\frac{kc^2}{8}G\right| - \left|\frac{C\Delta}{2}\right| = \frac{kC^2}{8}|G| - \frac{C}{2}.$$

Incorporating inequality (19) in above, we further get

$$\frac{2}{k|G|} \ge \left|\frac{C\Delta}{2} - \frac{kC^2}{8}G\right| \ge \frac{kC^2}{8}|G| - \frac{C}{2}. \tag{21}$$

Inequality (20) implies that

$$\frac{2}{k|G|} \ge \frac{C}{2} - \frac{kC^2}{8}|G|$$
$$\Rightarrow (k|G|C - 2)^2 + 12 \ge 0$$

which is trivially true.

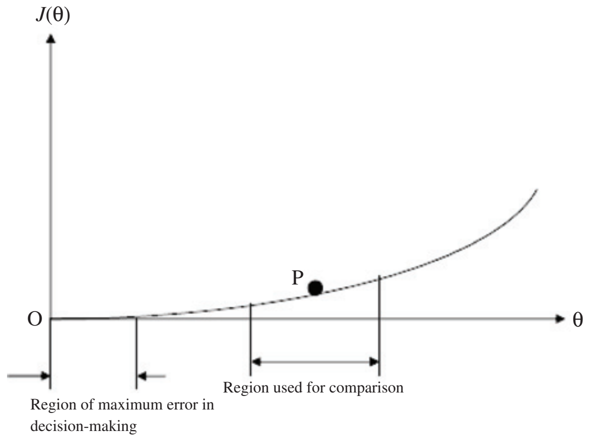

$J(\theta)$
P
O
θ
Region used for comparison
Region of maximum error in decision-making

Fig. 8. Sample fitness landscape for studying the computational chemotaxis.

From inequality (21), we get

$$\frac{2}{k|G|} \geq \frac{kC^2}{8}|G| - \frac{C}{2}$$
$$\Rightarrow k^2|G|^2C^2 - 4k|G|C - 16 \leq 0$$
$$\Rightarrow -\frac{2}{k|G|}(\sqrt{5}-1) \leq C \leq \frac{2}{k|G|}(1+\sqrt{5}).$$

But since $C > 0$

$$0 < C \leq \frac{2}{k|G|}(1+\sqrt{5}). \tag{22}$$

Now, let us assume within our domain of analysis that $|G|_{\max}$ is the maximum possible magnitude of gradient. The $2/(k|G|)(1+\sqrt{5})$ term is minimized when $|G| = |G|_{\max}$. Our analysis is valid if the chemotactic step size is less than or equal to this minimum value, i.e., $2/(k|G|_{\max})(1+\sqrt{5})$. So we define maximum allowable value of chemotactic step size as $C_{\max} = 2/(k|G|_{\max})(1+\sqrt{5})$. If $|G|_{\max}$ is large, the maximum allowable step size almost vanishes making our analysis invalid for moderately small values of step size. From this consideration also, we should restrict the domain of analysis to within the region with moderate value of gradient.

D. Chemotaxis and the Classical Gradient Decent Search

From (12) of Section IV-A, we get

$$V_b = -\frac{kC^2}{8}G + \frac{C.\Delta}{2} \quad \Rightarrow \frac{d\theta}{dt} = -\alpha' G + \beta' \tag{23}$$

where α' is $(-kC^2)/8$ and β' is $(C\Delta)/2$. The classical gradient descent search algorithm is given by the following dynamics in single dimension [10]:

$$\frac{d\theta}{dt} = -\alpha.G + \beta \tag{24}$$

where α is the learning rate and β is the momentum. The similarity between (23) and (24) suggests that chemotaxis may be considered a modified gradient descent search, where α', which is a function of chemotactic step-size, can be identified as the learning rate parameter.

Already we have discussed in Section IV-C that the magnitude of gradient should be small within the region of our analysis. So we choose point P in the 1-D fitness landscape shown in Fig. 8 as the operating point for our analysis. For chemotaxis of BFOA, when G becomes very small, the gradient descent term $\alpha' G$ of (23) becomes ineffective. But the random search term $(C.\Delta)/2$ plays an important role in this context. From (23), considering $G \to 0$, we have

$$\frac{d\theta}{dt} = \frac{C.\Delta}{2} \neq 0. \tag{25}$$

So there is a convergence toward actual minima. Fig. 8 shows a region on the fitness landscape with very a small value of gradient. The random search or momentum term $(C.\Delta)/2$ in the RHS of (12) provides an additional feature to the classical gradient descent search. When the gradient becomes very small, the random term dominates over gradient decent term and the bacterium changes its position. But the random search term may lead to change in position in the direction of increasing objective function value. If it happens, then again the magnitude of the gradient increases and dominates the random search term.

E. Oscillation Problem: Need for Adaptive Chemotaxis

If magnitude of the gradient decreases consistently near the optima or very close to the optima, $\alpha' G$ of (23) becomes comparable to β. Then, gradually β becomes dominant. When $|G| \to 0, |(d\theta)/(dt)| \approx |\beta| = |(C\Delta)/2| = C/2 \because |\Delta| = 1$. Let us assume the bacterium has reached close to the optimum. But since we obtain $|(d\theta)/(dt)| = C/2$, the bacterium does not stop taking chemotactic steps but oscillates about the optima. This crisis can be remedied if step size C is made adaptive according to the following relation:

$$C = \frac{|J(\theta)|}{|J(\theta)| + \lambda} = \frac{1}{1 + \lambda/|J(\theta)|} \tag{26}$$

where λ is a positive constant. The choice of a suitable value for λ is discussed in the next section. Here we have assumed that the global optimum of the cost function is 0. Thus from (26), if $J(\theta) \to 0$, then $C \to 0$. So there would be no oscillation if the bacterium reaches optima because the random search term vanishes as $C \to 0$. The functional form given in (26) causes C to vanish near the optima. On the other hand, from (26) we observe that when $J(\theta)$ is large, $\lambda/(|J(\theta)|) \to 0$ and consequently $C \to 1$.

The adaptation scheme presented in (26) has an important physical significance. If the magnitude of objective function is large for an individual bacterium, it is in the vicinity of a noxious substance. It will then try to move to a place with better nutrient concentration by taking large steps. On the other hand, when the bacterium is in a nutrient-rich zone, i.e., with small magnitude of the objective function value, it tries to retain its position. Naturally, its step size becomes small.

F. Adaptive Chemotaxis for Avoiding the Lock in State

Let us consider an even function $J(\theta)$ (as shown in Fig. 9), which has its minima at $\theta = 0$ and its minimum value equal

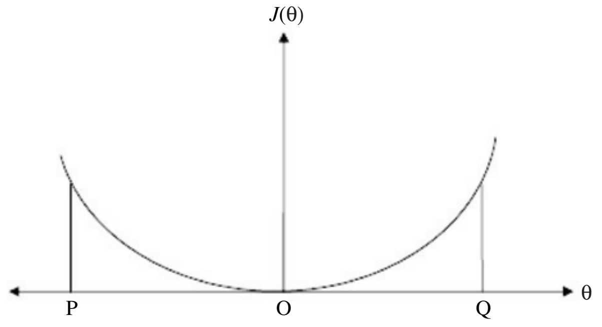


Fig. 9. Fitness landscape for the function $J(\theta) = \theta^2$.

to 0. Let us also assume the function is increasing in the interval $[0, \varphi]$ and decreasing in $[-\varphi, 0]$ (e.g., $J(\theta) = \theta^2$ is an even function where it is increasing in the interval $(0, \infty)$ and decreasing in $(-\infty, 0)$ so in this case $\varphi \to \infty$). A special case of stagnation may occur within the region $(-\varphi, \varphi)$. We here refer to this situation as "lock in." The lock in condition arises when a bacterium has reached somewhat near the optima of a function and its further movements are not possible due to the comparatively large step size.

Example 1: Suppose we have to minimize a 1-D function $J(\theta) = \theta^2$. A plot of the function has been provided in Fig. 9.

In Fig. 9, let $|\mathrm{PO}| = |\mathrm{QO}| = |\theta|$. We also assume that he bacterium is currently at the position $\theta = \theta$, i.e., it is at Q. Now, in classical chemotaxis three cases may arise as described below:

Case I: Let step size $C = 2\,|\theta|$. Then, for $\Delta = -1$, the bacterium should move to P. But as in this case its objective function value remains same, it does not come to P but stays at Q [as $J(\theta) = J(-\theta) = \theta^2$]. As $\Delta = 1$ tries to shift the bacterium to the right (where the objective function value increases again), it again stays at Q. Hence the bacterium gets trapped at Q.

Case II: Let $C > 2\,|\theta|$. In that case, the bacterium remains immobile for both values of Δ. Here step size is constant and greater than $2 \times |\theta|$. If the bacterium moves in any one of the two directions, the value of the objective function increases. So, bacterium is trapped.

Case III: Let $C < 2\,|\theta|$. In this case, bacterium will move to some point in the left of origin. However, C is fixed (say, 0.5). So, after certain iterations any one of Case II and I must arise.

The situation of the bacterium in these three cases is depicted in Fig. 10.

Now consider the situation where the step size has been adapted according to (26). Then we have $C = (\theta^2)/(\theta^2 + \lambda)$. The lock in states never occur if, for all possible values, of θ

$$\begin{aligned} C &< 2|\theta| \\ \Rightarrow \frac{|\theta|^2}{|\theta|^2 + \lambda} &< 2|\theta| \\ \Rightarrow |\theta|^2 &< 2|\theta|(|\theta|^2 + \lambda) \quad \left[\because |\theta|^2 + \lambda > 0\right] \\ \Rightarrow \lambda &> \frac{|\theta|}{2} - |\theta|^2. \end{aligned}$$

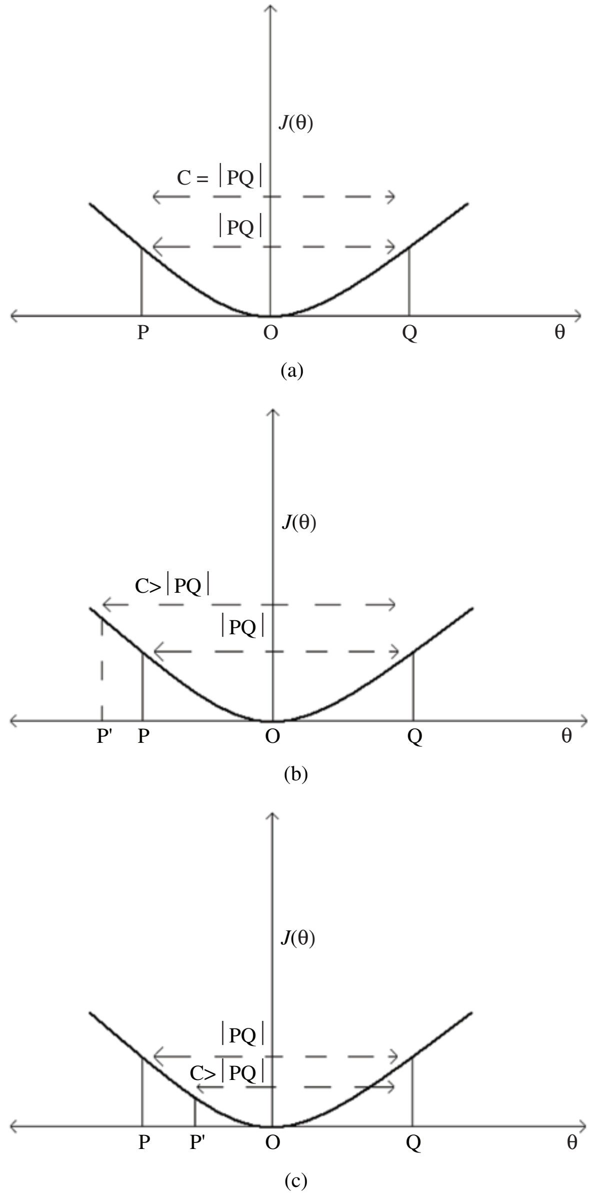


Fig. 10. Situation of a bacterium cell near the global optimum in classical chemotaxis: (a) Case I. (b) Case II. (c) Case III.

Let us consider $f(|\theta|) = (|\theta|)/2 - |\theta|^2$. The maximum value of the function is obtained when $(df(|\theta|))/(d|\theta|) = 0 \Rightarrow |\theta| = 1/4$. Putting $|\theta| = 1/4$, we get

$$\arg\max_{\theta} f(|\theta|) = f\left(\frac{1}{4}\right) = \frac{1}{16}.$$

Hence, for all values of θ, if $\lambda > f(|\theta|) \Rightarrow \lambda > (1/16)$, no trapping or sustained oscillations of the bacterium cell will arise near the global optimum. In this case, the bacterium cell will follow a trajectory as depicted in Fig. 11.

Next we provide a brief comparison between BFOA and the proposed ABFOA over the 1-D objective function $J(\theta) = \theta^2$. Each algorithm uses only a single bacterium, and in both the cases it is initialized at $\theta(0) = 6.0$. We have taken $\lambda > (1/16)$ to avoid lock in. The results of five iterations are tabulated in Table I and the convergence characteristics have been graphically presented in Fig. 12. The iteration signifies

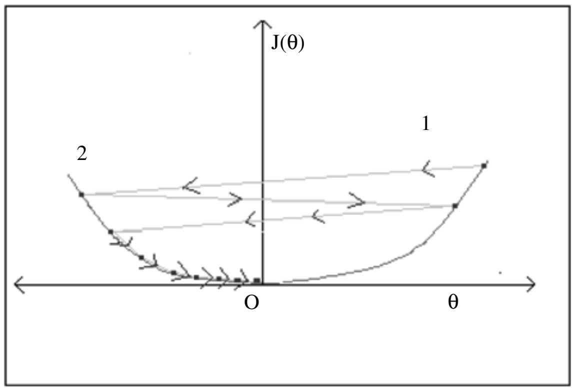


Fig. 11. Convergence toward the global optima for adaptive step size in chemotaxis.

TABLE I
VARIATION OF BACTERIUM POSITION Θ WITH CHEMOTACTIC STEPS FOR ADAPTIVE STEP SIZE *C*

Chemotactic step number	According to BFOA	According to ABFOA1
	θ	θ
1	0.600000	0.600000
2	0.170000	-0.175862
3	0.170000	0.053353
4	0.170000	0.053353
5	0.170000	0.026712

the chemotactic step number in this case. We can observe that, due to its constant step size, the BFOA-bacterium stops before reaching the optima. For ABFOA, the bacterium adapts the step size according to objective function value and gradually nears the optima. We get a better quality of the final solution in this case.

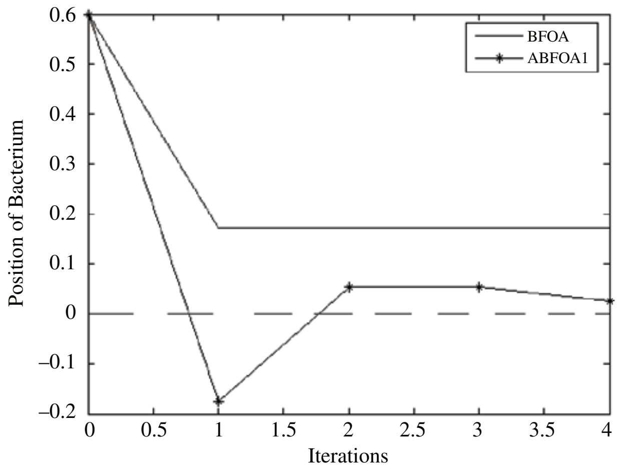


Fig. 12. Variation of bacterial position θ with time near the global optima for classical and adaptive chemotaxis.

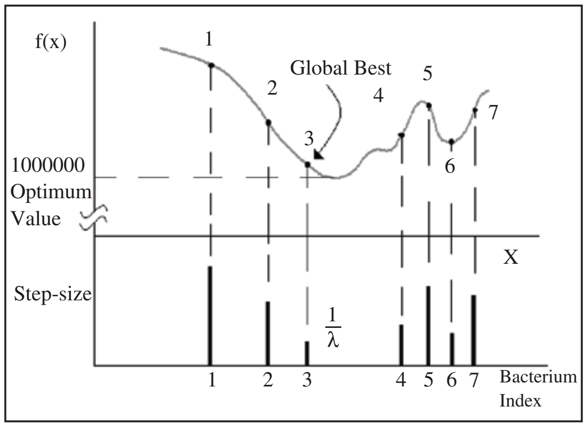


Fig. 13. Objective function with optimum value much greater than zero and a group of seven bacteria scattered over the fitness landscape. Their step heights are also shown.

G. Special Case

If the optimum value of the objective function is not exactly zero, the step size adapted according to (26) may not vanish near optima. The step size would shrink if the bacterium come closer to the optima, but it may not approach zero always. To get faster convergence for such functions it becomes necessary to modify the adaptation scheme. The use of gradient information in the adaptation scheme, i.e., making the step size a function of the function gradient [say, $C = C(J(\theta), G)$] may not be practical enough, because in real-life optimization problems, we often deal with discontinuous and nondifferentiable functions. Note that algorithms like standard gradient methods, quasi-Newton methods [33], Levenbarg–Marquardt algorithm [18], etc. depend on the use of an explicit analytical representation of the first- or second-order derivative, something that is not needed by a foraging or genetic algorithm. Even the classical BFOA uses only an approximation of the gradient and not the analytical gradient information. Thus, in order to keep BFOA a general black-box optimizer, our adaptive scheme should be a more general one, performing satisfactorily over discontinuous as well as nondifferentiable objective functions. Therefore we propose an alternative adaptation strategy in the following way:

$$C = \frac{|J(\theta) - J_{best}|}{|J(\theta) - J_{best}| + \lambda}. \tag{27}$$

J_{best} is the objective function value for the globally best bacterium (one with the lowest value of the objective function). $|J(\theta) - J_{best}|$ is the deviation in fitness value of an individual bacterium from the global best. Expression (27) can be rearranged to give

$$C = \frac{1}{1 + \frac{\lambda}{|J(\theta) - J_{best}|}}. \tag{28}$$

If a bacterium is far away from the global best, $|J(\theta) - J_{best}|$ would be large, making $C \approx 1 \because (\lambda)/(|J(\theta) - J_{best}|) \to 0$. On the other hand, if another bacterium is very close to it, the step size of that bacterium will almost vanish because $|J(\theta) - J_{best}|$ becomes small and the denominator of (28) grows very large. This scenario is depicted in Fig. 13. In what follows, we shall refer to the BFOA with the adaptive scheme of (26) as ABFOA1 and the BFOA with the adaptation scheme described in (27) as ABFOA2.

Fig. 13 shows how the step size becomes large as the objective function value becomes larger for an individual bacterium. The bacterium with better function value will try to take a smaller step and retain its present position. For the best bacterium of the swarm, $|J(\theta) - J_{best}|$ is 0. Thus, from (27) its step size is $1/\lambda$, which is quite small. The adaptation scheme bears a physical significance, too. A bacterium located at a relatively less nutrient region of the fitness landscape will take large step sizes to attain better fitness; whereas another bacterium located at a location best in regard to nutrient content is unlikely to move much.

Note that the globally best solution vector (one which yields the lowest objective function value) is employed to change the other population members in some derivative-free optimization techniques such as PSO [19] and the differential evolution (in variants like DE/best/1, DE/target-to-best/1/bin, etc.) [34]. Our adaptation scheme in (27) is, however, based on the best objective function value and not on the global best position itself.

In this context, it is worth mentioning that the chemotactic step size C controls the amplitude of the step taken by a bacterium toward a random direction specified by the tumble vector Δ. Effectively, the role of C is somewhat similar to the mutation step size employed by evolutionary strategies (ESs) [35] and evolutionary programming (EP) [36]. In some self-adaptive variants of ES [37], [38], the mutation step size is evolved along with other search variables. The self-adaptation rules from ESs have also been incorporated into EP [39]. Some variants of EP incorporating the self-adaptation of variances of mutations have been proposed in literature [40], [41]. Recently, Lee and Yao [42] studied a new mutation scheme for EP based on the Lévy probability distribution. The Lévy mutation operator proposed by them can lead to a large mutation step size (i.e., large variance) and a large number of distinct values in evolutionary search, as compared to the classical Gaussian mutation. Our present adaptation schemes of the chemotactic step size follows from a simple differential equation-based model of the bacterial dynamics and differs substantially from the adaptation mechanisms commonly found in the ES or EP literature. However, it is interesting to investigate in future the adaptation of C and Δ based on Cauchy or Lévy probability density functions, taking inspirations from the works reported in [42], [43].

For real-world optimization problems, where the optimum value of the objective function is zero, the adaptation scheme of (26) works satisfactorily. But for functions that do not have a small optimum value, (27) should be used for better convergence. Note that neither of two proposed schemes contains the derivative of objective function, so they can be used for discontinuous and nondifferentiable functions as well.

V. Experiments and Results Over Benchmark Functions

This section presents an extensive comparison among the performances of two adaptive BFOA schemes (ABFOA1 and ABFOA2), the classical BFOA, the BSO algorithm, a standard real-coded GA, and one of the state-of-the-art variants of the PSO algorithm.

A. Numerical Benchmarks

Our test suite includes 10 well-known benchmark functions [43] of varying complexity. In Table II, p represents the number of dimensions and we used $p = 15$, 30, 45, and 60 for functions f_1 to f_7, while functions f_8 to f_{10} are 2-D. The first function is unimodal with only one global minimum. The others are multimodal with a considerable number of local minima in the region of interest. Table II summarizes the initialization and search ranges used for all the functions. An asymmetrical initialization procedure has been used in this research following the work reported in [44].

B. Algorithms Used for the Comparative Study and Their Parametric Setup

1) BFOA and its Adaptive Variants: The original BFO and the two adaptive BFOA schemes employ the same parametric setup, except with the difference that the chemotactic step sizes in ABFOA1 and ABFOA2 have been made adaptive according to (25) and (26), respectively. After performing a series of hand-tuning experiments, we found that keeping $\lambda = 4000$ provides considerably good results for both the adaptive schemes over all benchmark functions considered here. The chemotactic step-size $C(i)$ was kept at 0.1 in the classical BFOA. Rest of the parameter settings that were kept same for these algorithms have been provided in Table III. In order to make the comparison fair enough, all runs of the three BFOA variants start from the same initial population over all the problem instances.

2) HPSO-TVAC Algorithm: PSO [19, 45] is a stochastic optimization technique that draws inspiration from the behavior of particles, the boids method of Reynolds, and socio-cognition. In the classical PSO a population of particles is initialized with random positions $\vec{X}_i$ and velocities $\vec{V}_i$, and a function f is evaluated using the particle's positional coordinates as input values. In a D-dimensional search space, $\vec{X}_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]^T$ and $\vec{V}_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]^T$.

Positions and velocities are adjusted, and an objective function is evaluated with the new coordinates at each time step. The fundamental velocity and position update equations for the dth dimension of the ith particle in the swarm may be given as

$$v_{id}(t+1) = \omega.v_{id}(t) + C_1.\varphi_1.(P_{id} - x_{id}(t)) + C_2.\varphi_2.(P_{gd} - x_{id}(t)) \tag{29a}$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1). \tag{29b}$$

The variables φ_1 and φ_2 are random positive numbers drawn from a uniform distribution and restricted to an upper limit $\varphi_{\max}$ (usually equal to 2), which is a parameter of the system. C_1 and C_2 are called acceleration coefficients, whereas ω is known as the inertia weight. P_{id} is dth component of the personal best solution found so far by an individual particle, while P_{gd} represents dth element of the globally best particle found so far in the entire community.

Ratnaweera *et al.* [46] recently suggested a parameter automation strategy for PSO where the cognitive component is reduced and the social component is increased [by varying

TABLE II
DESCRIPTION OF THE BENCHMARK FUNCTIONS USED

Function	Mathematical representation	Range of search	Theoretical optima
Sphere function (f_1)	$f_1(\vec{x}) = \sum_{i=1}^{p} x_i^2$	(–100, 100)^p	$f_1(\vec{0}) = 0$
Rosenbrock (f_2)	$f_2(\vec{x}) = \sum_{i=1}^{p-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	(–100, 100)^p	$f_2(\vec{1}) = 0$
Rastrigin (f_3)	$f_3(\vec{x}) = \sum_{i=1}^{p} [x_i^2 - 10\cos(2\pi x_i) + 10]$	(–10, 10)^p	$f_3(\vec{0}) = 0$
Griewank (f_4)	$f_4(\vec{x}) = \frac{1}{4000} \sum_{i=1}^{p} x_i^2 - \prod_{i=1}^{p} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	(–600, 600)^p	$f_4(\vec{0}) = 0$
Ackley (f_5)	$f_5(\vec{X}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) -\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos 2\pi x_i\right) + 20 + e$	$(-32, 32)^p$	$f_5(\vec{0}) = 0$
Step (f_6)	$f_6(\vec{X}) = \sum_{i=1}^{p} (\lfloor x_i + 0.5 \rfloor)^2$	(–100, 100)^p	$f_6(\vec{p}) = 0$, $-\frac{1}{2} \le p_i < \frac{1}{2}$
Schwefel's Problem 2.22 (f_7)	$f_7(\vec{X}) = \sum_{i=1}^{p} \lvert x_i \rvert + \prod_{i=1}^{p} \lvert x_i \rvert$	(–500, 500)^p	$f_7(\vec{0}) = 0$
Shekel's Fox-holes (f_8)	$f_8(\vec{x}) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}\right]^{-1}$	(–65.536, 65.536)^2	$f_8(-32, -32) = 0.998$
Six-Hump Camel-Back function (f_9)	$f_9(\vec{X}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^6$	$(-5, 5)$	$f_9(0.08983, -0.7126) = f_9(-0.08983, 0.7126) = -1.0316285$
Goldstein–Price function (f_{10})	$f_{10}(\vec{x}) = \{1 + (x_0 + x_1 + 1)^2(19 - 14x_0 + 3x_0^2 - 14x_1 - 6x_0x_1 + 3x_1^2)\}\{30 + (2x_0 - 3x_1)^2 \times (18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2)\}$	$(-2, 2)^2$	$f_{10}(0, -1) = 3$

TABLE III
COMMON PARAMETER SETUP FOR BFOA AND ADAPTIVE BFOA (ABFOA)

S	N_c	N_s	N_{ed}	N_{re}	p_{ed}	$d_{attractant}$	$w_{attractant}$	$w_{repellant}$	$h_{repellant}$	λ
100	100	12	4	16	0.25	0.1	0.2	10	0.1	400

the acceleration coefficients C_1 and C_2 in (29a)] linearly with time. They suggested another modification, named "self-organizing hierarchical particle swarm optimizer," in conjunction with the previously mentioned time varying acceleration coefficients (HPSO-TVAC). In this method, the inertial velocity term is kept at zero and the modulus of the velocity vector is reinitialized to a random velocity, known as "re-initialization velocity," whenever the particle gets stagnant ($v_{id} = 0$) in some region of the search space. This way, a series of particle swarm optimizers are generated automatically inside the main particle system according to the behavior of the particles in the search space, until some stopping criterion is met. Here we compare this state-of-the-art version of PSO with the adaptive BFOA schemes. The parametric setup for HPSO-TVAC follows the work reported in [46]. The re-initialization velocity is kept proportional to the maximum allowable velocity $\vec{V}_{\max}$. We fixed the number of particles equal to 40 and the inertia weight $\omega = 0.794$. C_1 was linearly increased from 0.35 to 2.4, while C_2 was allowed to decrease linearly from 2.4 to 0.35. Finally, $\vec{V}_{\max}$ was set at $\vec{X}_{\max}$.

3) Real-Coded GA: In this research, we used a standard real-coded GA (also known as evolutionary algorithm or EA [47]) that was previously found to work well on real-world problems [48]. The EA works as follows: First, all individuals are randomly initialized and evaluated according to a given objective function. Afterwards, the following process will be executed as long as the termination condition is not fulfilled: Each individual is exposed to either mutation or recombination (or both) operators with probabilities p_m and p_c, respectively. The mutation and recombination operators used are Cauchy mutation with an annealing scheme and arithmetic crossover, respectively. Finally, tournament selection (of size 2) [47] is applied between each pair of individuals to remove the least fit members of the population. The Cauchy mutation operator is similar to the well-known Gaussian mutation operator, but the Cauchy distribution has thick tails that enable it to generate considerable changes more frequently than the Gaussian distribution. The Cauchy distribution may be presented as

$$C(x,\alpha,\beta) = \frac{1}{\beta\pi\left(1+\left(\frac{x-\alpha}{\beta}\right)^2\right)} \tag{30}$$

where $\alpha \geq 0$, $\beta > 0$, and $-\infty < x < \infty$. An annealing scheme is employed to decrease the value of β as a function of the elapsed number of generation t while α is fixed to 0. In this paper, we used the following annealing function:

$$\beta = \frac{1}{1+t}. \tag{31}$$

In arithmetic crossover, the offspring is generated as a weighted mean of each gene of the two parents, i.e.,

$$offspring_i = r.parent1_i + (1-r).parent2_i. \tag{32}$$

The weight r is determined by a random value between 0 and 1. Here we fixed the population size at 100, $p_m = 0.9$, and $p_c = 0.7$, for all the problem instances.

4) Bacterial Swarm Optimization: Biswas *et al.* [27] proposed a hybrid optimization technique that synergistically couples BFOA with PSO. The algorithm, referred to as the BSO, 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 the best of both the worlds. In BSO, after undergoing a chemotactic step, each bacterium also gets mutated by a PSO operator. In this phase, the bacterium is stochastically attracted toward the globally best position found so far in the entire population at the current time and also toward its previous heading direction. The PSO operator uses only the globally best position found by the entire population to update the velocities of the bacteria and eliminates term involving the personal best position, as the local search in different regions of the search space is already taken care of by the chemotactic operator of BFOA. The parametric setup for the algorithm was kept exactly same as described in [27]. For the PSO operator we choose $\omega = 0.8$ and $C_2 = 1.494$, while for the BFO operators the parameter values were kept as described in Table III.

C. Simulation Strategy

The comparative study presented in this paper focuses on the following performance metrics: 1) the quality of the final solution; 2) the convergence speed [measured in terms of the number of fitness function evaluations (FEs)]; and 3) the frequency of hitting the optima. Fifty independent runs of each of the algorithms were carried out, and the average and the standard deviation of the best-of-run values were recorded.

For a given function of a given dimension, 50 independent runs of each of the six algorithms were executed, and the average best-of-run value and the standard deviation were obtained. Different maximum numbers of function evaluations (FEs) were used according to the complexity of the problem. For benchmarks f_1 to f_7, the stopping criterion was set as reaching an objective function value of 0.001. However, for f_8, f_9, and f_{10} the stopping criteria are fixed at 0.998, -1.0316, and 3.00, respectively. In order to compare the speeds of different algorithms, we note down the number of FEs an algorithm takes to converge to the optimum solution (within the given tolerance). A lower number of FEs corresponds to a faster algorithm. We also keep track of the number of runs of each algorithm that manage to converge within the prespecified error limit over each problem.

We used t-tests to compare the means of the results produced by the best ABFOA scheme and the best of the other competitor algorithms over each problem. The t-test assumes that the data has been sampled from a normally distributed population. From the concepts of the central limit theorem, one may note that as the sample sizes increase, the sampling distribution of the mean approaches a normal distribution regardless of the shape of the original population. A sample size of around 50 allows the normality assumptions conducive for performing the t-tests [49].

D. Empirical Results

Table IV compares the algorithms on the quality of the best solutions obtained. The mean and the standard deviation (within parentheses) of the best-of-run solution for 50 independent runs of each of the 10 algorithms are presented in Table IV. Note that in this table, if all the runs of a particular algorithm converge to or below the prespecified objective function value (0.001 for f_1 to f_6; 0.998, -1.0316, and 3.00 for f_8, f_9, and f_{10}, respectively) within the maximum number of FEs, then we report this threshold value as the mean of 50 runs. Missing values of standard deviation in these cases indicate a zero standard deviation. Table V shows results of t-tests between the best algorithm and the second best in each case (standard error of difference of the two means, 95% confidence interval of this difference, the t-value, and the two-tailed P value). For all cases in Table III, the sample size was 50 and number of degrees of freedom was 98. This table covers only those cases for which a single algorithm achieves the best accuracy of final results. Table VI shows, for all test functions and all algorithms, the number of runs (out of 50) that managed to find the optimum solution (within the given tolerance) and also the average number of FEs taken to reach the optima along with the standard deviation (in parentheses). Missing values of standard deviation in this table also indicate a zero standard deviation. The entries marked as zero in this table indicate that no runs of the corresponding algorithm could manage to converge within the given tolerance in those cases. In all the tables, the entries marked in bold represent the comparatively best results. The convergence characteristics of six most difficult benchmarks have been provided in Fig.14 for the median run of each algorithm (when the runs were ranked according to their final accuracies). Each graph shows how the objective function value of the best individual in a population changes with increasing number of FEs. Some of the illustrations have been omitted in order to save space.

TABLE IV
AVERAGE AND THE STANDARD DEVIATION (IN PARENTHESIS) OF THE BEST-OF-RUN SOLUTION FOR 50 INDEPENDENT RUNS TESTED ON TEN BENCHMARK FUNCTIONS

Func	Dim	Maximum no. of FEs	Mean best value (standard deviation)					
			BFOA	HPSO-TVAC	EA	BSO	ABFOA1	ABFOA2
f_1	15	5×10^4	0.0086 (0.0044)	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	30	1×10^5	0.084 (0.0025)	0.065 (0.0534)	0.036 (0.001)	0.056 (0.0112)	**0.022 (0.00625)**	0.044 (0.0721)
	45	5×10^5	0.776 (0.1563)	0.383 (0.05564)	0.257 (0.0323)	0.354 (0.2239)	**0.208 (0.0664)**	0.419 (0.2096)
	60	1×10^6	1.728 (0.2125)	1.364 (0.5136)	0.6351 (0.0298)	0.775 (0.4291)	**0.427 (0.1472)**	0.632 (0.5747)
f_2	15	5×10^4	36.595 (28.1623)	94.472 (75.8276)	14.756 (10.5552)	**0.673 (0.1454)**	11.561 (2.355)	15.4931 (4.3647)
	30	1×10^5	58.216 (14.3254)	706.263 (951.9533)	31.738 (3.6452)	15.471 (2.655)	**4.572 (3.0631)**	6.748 (2.6625)
	45	5×10^5	96.873 (26.136)	935.2601 (1102.352)	67.473 (16.3526)	30.986 (4.3438)	**24.663 (10.8644)**	39.736 (30.6261)
	60	1×10^6	154.705 (40.1632)	1264.287 (1323.5284)	109.562 (34.7275)	**76.647 (24.5281)**	91.257 (32.6283)	84.6473 (53.2726)
f_3	15	5×10^4	10.4521 (5.6632)	9.467 (3.726)	0.4981 (0.0376)	**0.2632 (0.2348)**	0.3044 (0.6784)	2.6573 (0.0372)
	30	1×10^5	17.5248 (9.8962)	34.837 (10.128)	3.797 (0.8241)	13.7731 (3.9453)	**2.5372 (0.3820)**	2.9823 (0.5719)
	45	5×10^5	32.9517 (10.0034)	46.332 (22.4518)	8.536 (2.7281)	18.9461 (7.7075)	**6.0236 (1.4536)**	8.1121 (4.3625)
	60	1×10^6	41.4823 (17.6639)	58.463 (66.4036)	12.0922 (4.5631)	10.2266 (2.8942)	**8.3343 (0.2917)**	9.4637 (6.7921)
f_4	15	5×10^4	0.2812 (0.0216)	0.0564 (0.025810	0.05198 (0.00487)	0.1741 (0.097)	**0.0321 (0.02264)**	0.05113 (0.02351)
	30	1×10^5	0.3729 (0.0346)	0.2175 (0.1953)	0. 2684 (0.3616)	0.2565 (0.1431)	**0.1914 (0.0117)**	0.2028 (0.1532)
	45	5×10^5	0.6351 (0.0522)	0.4748 (0.4561)	0.3732 (0.0971)	0.5678 (0.236)	0.3069 (0.0526)	**0.3065 (0.0923)**
	60	1×10^6	0.8324 (0.0764)	0.7462 (0.5521)	0.6961 (0.4737)	0.7113 (0.097)	**0.5638 (0.3452)**	0.6074 (0.5731)
f_5	15	5×10^4	0.9332 (0.0287)	0.1217 (0.0125)	**0.001482 (0.00817)**	0.1025 (0.00347)	0.7613 (0.0542)	0.6757 (0.2741)
	30	1×10^5	2.3243 (1.8833)	0.5684 (0.1927)	0.6059 (0.3372)	0.5954 (0.1246)	**0.5038 (0.5512)**	0.7316 (0.6745)
	45	5×10^5	3.4564 (3.4394)	0.9782 (0.2029)	**0.9298 (0.7631)**	1.0383 (0.2542)	1.5532 (0.1945)	1.3672 (0.4618)
	60	1×10^6	4.3247 (1.5613)	2.0293 (3.7361)	1.8353 (1.4635)	1. 9166 (0.536)	**1.7832 (0.4581)**	1.9272 (0.7734)
f_6	15	5×10^4	0.0400 (0.00283)	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
	30	1×10^5	2.0802 (0.00342)	0.7752 (0.4531)	**0.001**	0.4852 (0.28271)	**0.001**	**0.001**
	45	5×10^5	14.7328 (3.2827)	13.8478 (2.5673)	6.8825 (0.6471)	4.2832 (0.6476)	**1.1372 (0.8539)**	1.2062 (0.5915)
	60	1×10^6	19.8654 (4.8271)	15.8272 (2.5362)	12.6574 (0.4321)	17.6664 (0.3762)	**2.3462 (0.3474)**	6.1224 (1.5365)
f_7	15	5×10^4	2.8271 (0.3029)	1.6645 (0.4198)	**0.001**	0.8817 (0.6362)	**0.001**	0.0442 (0.1096)
	30	1×10^5	4.6354 (2.7753)	2.4861 (2.3375)	0.0642 (0.7681)	0.9043 (0.4186)	**0.0084 (0.00037)**	0.0405 (0.0252)
	45	5×10^5	9.4563 (10.2425)	5.5674 (0.3526)	6.2452 (2.3724)	1.7828 (0.4652)	**0.0484 (0.0335)**	0.0563 (0.04634)
	60	1×10^6	16.4638 (12.40940	10.6273 (12.4938)	11.5748 (9.3526)	6.4482 (7.4432)	**0.8256 (0.2282)**	1.4643 (0.9435)
f_8	2	1×10^5	1.056433 (0.01217)	0.9998323 (0.00537)	0.9998329 (0.00382)	0.9998017 (0.00825)	0.9998564 (0.00697)	**0.9998004 (0.00481)**
f_9	2	1×10^5	−0.925837 (0.000827)	−1.029922 (1.382)	−1.031149 (2.527)	−1.031242 (0.00759)	−1.03115 (0.0242)	−1.031593 **(0.000472)**
f_{10}	2	1×10^5	3.656285 (0.109365)	3.1834435 (0.2645)	3.146090 (0.06237)	3.443712 (0.007326)	3.572012 (0.00093)	**3.00000**

TABLE V
RESULTS OF t-TESTS ON THE DATA OF TABLE IV

Fn, Dim.	Std. Err	t	95% Confidence interval	Two-tailed P
f_1, 30	0.001	11.3352	−0.0164510 to −0.0115490	< 0.0001
f_1, 45	0.010	4.6924	0.028277 to 0.069723	< 0.0001
f_1, 60	0.021	9.7978	0.165951 to 0.250249	< 0.0001
f_2, 15	0.334	32.6299	−11.550180 to −10.225820	< 0.0001
f_2, 30	0.573	19.0122	9.761375 to 12.036625	< 0.0001
f_2, 45	1.655	3.8212	3.039275 to 9.606725	= 0.0002
f_2, 60	8.294	0.9646	−24.459658 to 8.459058	= 0.3371
f_3, 15	0.102	0.4058	−0.242671 to 0.160271	0.6858
f_3, 30	0.128	9.8071	1.004880 to 1.514720	< 0.0001
f_3, 45	0.437	5.7471	1.644868 to 3.379932	< 0.0001
f_3, 60	0.411	4.5999	1.075939 to 2.708661	< 0.0001
f_4, 15	0.003	6.0702	0.0133808 to 0.0263792	< 0.0001
f_4, 30	0.028	0.9433	−0.028808 to 0.081008	= 0.3479
f_4, 45	0.019	3.5205	−0.104298 to −0.029102	0.0007
f_4, 60	0.083	0.0929	−0.172197 to 0.156797	= 0.9262
f_5, 15	0.039	17.3854	−0.75117727 to −0.59725873	< 0.0001
f_5, 30	0.011	5.6430	−0.087318 to −0.041882	< 0.0001
f_5, 45	0.126	3.4675	−0.687723 to −0.187077	= 0.0008
f_5, 60	0.217	0.2402	−0.378277 to 0.482477	= 0.8107
f_6, 45	0.092	20.5597	1.699894 to 2.063106	0.0001
f_6, 60	0.061	42.7282	2.4899255 to 2.7324745	0.0001
f_7, 30	0.109	0.5137	−0.1597643 to 0.2713643	= 0.6086
f_7, 45	0.066	26.2949	1.603505 to 1.865295	< 0.0001
f_7, 60	1.053	5.3390	3.532713 to 7.712487	< 0.0001
f_8	0.001	0.0010	−0.002678813 to 0.002681413	0.9992
f_9	0.000	2.7769	0.00010016 to 0.00060184	0.0066
f_{10}	0.009	16.5626	0.12858610 to 0.16359390	< 0.0001

VI. DISCUSSION ON THE RESULTS

From Table IV, it may be observed that the performance of both the adaptive variants remained consistently superior to that of the classical BFOA over all benchmark problems. A close inspection of Table IV also reveals that, out of 31 test cases, the adaptive BFOA schemes (ABFOA1 or ABFOA2 or both) outperformed all other contestant algorithms in 22 cases. It is also interesting to note from Table V that, out of these 22 benchmark instances, in 17 cases the difference between means of the ABFOA methods and other algorithms is statistically significant within a 95% confidence interval.

According to Table IV, EA and BSO remained the toughest competitors of the adaptive BFOA variants in most of the cases. The sphere function (f_1) is perhaps the easiest among all tested benchmarks. From Tables IV and VI, we find that for the 15-dimensional sphere 50 runs of all the algorithms converged to or below the prespecified objective function value of 0.001. Similar is the case for the step function (f_6) in 15 dimensions. BSO was found to yield better average accuracy (i.e., numerically larger average value of the function) than the proposed schemes over three cases (f_2 in 15 dimensions, f_2 in 60 dimensions, and f_3 in 15 dimensions). However, Table V indicates that, for functions f_2 in 60 dimensions and f_3 in 15 dimensions, the differences are not statistically significant. For functions f_7, f_8, and f_9, since the optima is not located at the origin, as expected, ABFOA2 with the second adaptation scheme performs better than ABFOA1 and the classical BFOA. Especially for function f_9 and f_{10}, the final average accuracy of ABFOA2 is significantly better (as evident from the two-tailed P-values in Table V) than all other algorithms. Only in two cases (f_5 in 15 dimensions and f_5 in 45 dimensions). But for remaining functions ABFOA1 outperforms ABFOA2 in most of the cases. The EA was found to outperform the adaptive BFO algorithms in a statistically meaningful way. We find that only in two cases (f_1 in 15 dimensions and f_6 in 15 dimensions) the HPSO-TVAC could yield comparable results with respect to the EA, BSO, and ABFOAs. We believe that the performance of this algorithm could be improved by judiciously tuning its parameters.

Table VI and Fig. 14 are indicative of the fact that the convergence behavior of the adaptive BFOAs has been considerably improved in comparison to that of their classical counterpart. From Table V, we note that in 24 problem instances (out of 31) not only do the ABFOAs produce most accurate results but they do so consuming the least amount of computational time (measured in terms of the number of FEs needed to converge). In addition, the frequency of hitting the optima is also greatest for ABFOAs over most of the benchmark problems covered here.

Since original BFOA and its adaptive variants start from the same intial population and use a common parametetric setup, the difference in their performance must have resulted from the use of adaptive chemotactic step height in ABFOAs. This observation also agrees with the simplified analytical treatment provided in Section III.

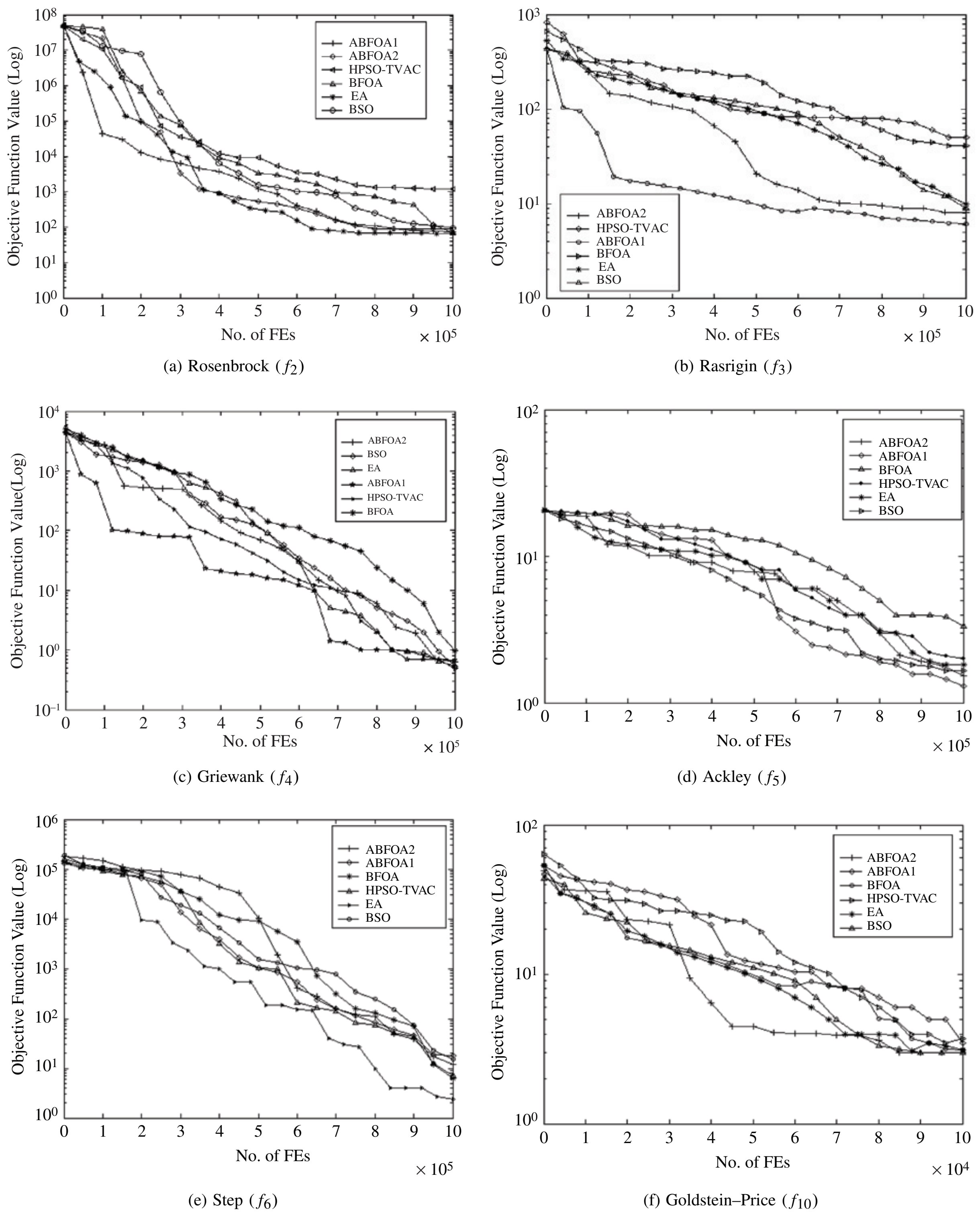


Fig. 14. Progress toward the optimum solution: (a)–(e) for dimension = 60 and (f) for dimension = 2.

VII. Application to Parameter Estimation for Frequency-Modulated (FM) Sound Waves

Frequency-modulated (FM) sound synthesis plays an important role in several modern music systems. This section describes an interesting application of the proposed ABFO algorithms to the optimization of parameters of an FM synthesizer. A few related works that attempt to estimate parameters of the FM synthesizer using GA can be found in [50], [51]. Here, we introduce a system that can automatically generate sounds similar to the target sounds. It consists of an FM synthesizer, an ABFOA core, and a feature extractor. The system architecture is shown in Fig. 15. The target sound is a *.wav* file. The ABFOA initializes a set of parameters, and the FM synthesizer generates the corresponding sounds. In the feature extraction step, the dissimilarities of features between

TABLE VI

NUMBER OF SUCCESSFUL RUNS, MEAN NUMBER OF FEs AND STANDARD DEVIATION (IN PARANTHESIS) REQUIRED TO CONVERGE TO THE THRESHOLD OBJECTIVE FUNCTION VALUE OVER THE SUCCESSFUL RUNS FOR FUNCTIONS f_1 TO f_{10}

Func	Dim.	No. of runs converging to the pre-defined objective function value, mean no. of FEs required and (standard deviation)					
		BFOA	HPSO-TVAC	EA	BSO	ABFOA1	ABFOA2
f_1	15	50, 16253.20 (445.34)	50, 12044.22 (610.298)	50, 13423.68 (341.827)	**50, 1544.34 (85.261)**	50, 1932.64 (140.492)	50, 4317.22 (310.298)
	30	37, 48931.5 (0.025)	42, 18298.21 (130.34)	44, 7364.32 (223.83)	42, 8367.86 (450.12)	**46, 3473.50 (346.22)**	38, 13296.46 (674.25)
	45	24, 172228.73 (6473.45)	45, 84712.34 (2552.34)	41, 36523.46 (2326.74)	39, 74782.63 (6638.93)	**43, 17832.65 (1423.45)**	32, 23876.65 (7731.63)
	60	8, 454563.25 (7653.44)	10, 676278.60 (10183.23)	31, 337822.63 (7198.45)	25, 167322.43 (0.4291)	**36, 88934.50 (4512.46)**	32, 72874.34 (6722.91)
f_2	15	0	0	1, 48585	**10, 47483.50 (2561.67)**	4, 16374.50 (6231.59)	0
	30	1, 93723	0	2, 76874.5 (9324.76)	4, 77563.75 (558.34)	**7, 87539.57 (4648.33)**	1, 80348
	45	0	0	0	**1, 127687**	0	0
	60	0	0	0	**1, 363727**	0	0
f_3	15	0	0	24, 17474.25 (2327.58)	**23, 28731 (7583.73)**	20, 33928.25 (5753.83)	13, 19823.56 (4100.67)
	30	0	0	4, 73483.50 (11142.76)	0	**16, 74722.52 (3432.67)**	5, 57382.60 (3423.73)
	45	0	0	1, 372833	0	**9, 175834.67 (3342.76)**	1, 46736.83
	60	0	0	0	0	**5, 478237.20 (22938.26)**	0
f_4	15	0	18, 37583.67 (7432.82)	16, 41029.75 (3732.68)	6, 14784.33 (4838.37)	**28, 26473.05 (3425.69)**	20, 29280.46 (4463.27)
	30	0	10, 75834.80 (4877.89)	6, 85734.46 (4000.03)	9, 81634.46 (4637.83)	24, 75834.46 (4637.83)	**23, 78583.37 (10093.35)**
	45	0	7, 292643.54 (2281.45)	4, 394852.75 (33621.38)	6, 475832.65 (1343.73)	**22, 137474.73 (4473.26)**	14, 302934.57 (5548.38)
	60	0	0	1, 543736	0	**14, 476375.43 (8636.55)**	10, 367482.60 (2386.43)
f_5	15	0	27, 38232.57 (4537.54)	**46, 18473.62 (2276.83)**	35, 27484.78 (7473.56)	23, 23789.67 (4839.57)	15, 24837.45 (4739.78)
	30	0	23, 74623.45 (3336.32)	24, 73832.68 (10298.56)	16, 84737.68 (4451.27)	**27, 54672.22 (6748.46)**	29, 84733.57 (3034.92)
	45	0	0.9782 (0.2029)	**18, 264723.57 (46.223)**	12, 472631.67 (4948.68)	20, 302862.60 (13741.34)	17, 383721.47 (17356.05)
	60	0	0	1, 634637	2, 737620.50 (3442.33)	**4, 607232.25 (34812.67)**	0
f_6	15	43, 6378.46 (394.35)	50, 16478.84 (425.32)	50, 33623.46 (364.38)	50, 46354.44 (2257.27)	**50, 10923.56 (3364.29)**	50, 22635.80 (1214.23)
	30	23, 84747.45 (3476.48)	16, 79534.32 (7904.52)	50, 68794.24 (6068.45)	39, 64726.32 (9830.51)	**50, 63778.40 (2385.31)**	50, 64532.64 (9336.46)
	45	0	0	34, 265732.58 (14527.35)	25, 748237.40 (4752.87)	**42, 205472.02 (8109.56)**	40, 284938.64 (7573.24)
	60	0	0	20, 684723.80 (13427.46)	0	**34, 475978.73 (5741.27)**	26, 584032.51 (4535.34)
f_7	15	34, 27243.44 (447.03)	32, 29583.49 (341.57)	50, 17263.92 (832.45)	31, 48374.34 (227.48)	**50, 16279.52 (723.47)**	43, 17563.46 (519.46)
	30	24, 64534.69 (4724.56)	30, 69203.67 (7311.46)	18, 73945.57 (3427.47)	27, 74932.33 (4825.45)	**40, 78473.50 (3412.67)**	37, 64722.59 (5174.38)
	45	0	0	0	23, 126377.65 (7106.74)	**34, 192935.37 (3691.62)**	25, 328372.63 (7148.42)
	60	0	0	0	0	**12, 542536.73 (2353.19)**	8, 632372.35 (12305.37)
f_8	2	38, 32928.14 (4118.982)	23, 26843.92 (6323.372)	50, 30272.74 (3642.289)	50, 19823.70 (4249.392)	44, 20354.77 (326.84)	**50, 12928.50 (15.749)**
f_9	2	46, 25374.87 (2643.839)	40, 31928.70 (1434.327)	50, 28372.74 (325.673)	50, 66290.80 (7553.388)	26, 87812.83 (409.54)	**50, 24883.78 (3172.827)**
f_{10}	2	35, 139584.44 (2563.378)	30, 347285.80 (3382.229)	42, 129372.87 (8742.093)	50, 126574.64 (6833.189)	40, 578732.05, (3884.94)	**50, 50039.60 (481.278)**

TABLE VII
AVERAGE AND THE STANDARD DEVIATION OF THE BEST-OF-RUN SOLUTION FOR 50 RUNS OF SIX ALGORITHMS ON THE FREQUENCY MODULATOR SYNTHESIS DESIGN PROBLEM. EACH ALGORITHM WAS RUN FOR 10^6FES

Mean best-of-run solution (std. deviation)					
BFOA	HPSO-TVAC	EA	BSO	ABFOA1	ABFOA2
2.74849 (0.8314)	0.76535 (0.1154)	0.0154 (0.00264)	0.75932 (0.2735)	**0.00365 (0.000851)**	0.00451 (0.00163)

TABLE VIII
NO. OF SUCCESSFUL RUNS, MEAN NUMBER OF FES AND STANDARD DEVIATION (IN PARANTHESIS) REQUIRED TO CONVERGE TO THE THRESHOLD FITNESS OVER THE SUCCESSFUL RUNS FOR THE FREQUENCY MODULATOR SYNTHESIS DESIGN PROBLEM

No. of runs converging to the predefined objective function value, mean no. of FEs required, and (standard deviation)					
BFOA	HPSO-TVAC	EA	BSO	ABFOA1	ABFOA2
0	6, 579534.32 (7904.52)	29, 868794.24 (63068.45)	23, 864726.32 (12830.51)	**42, 113778.40 (9385.31)**	36, 164532.64 (9336.46)

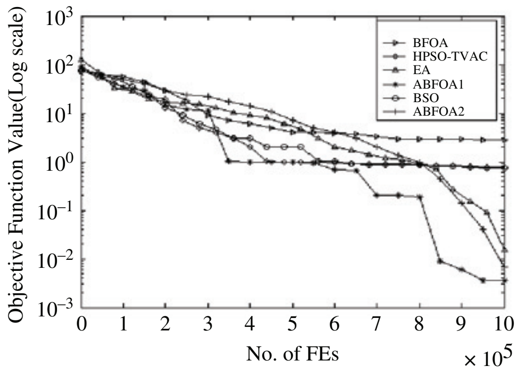


Fig. 15. Progress toward the optimum solution for the frequency modulator synthesis problem.

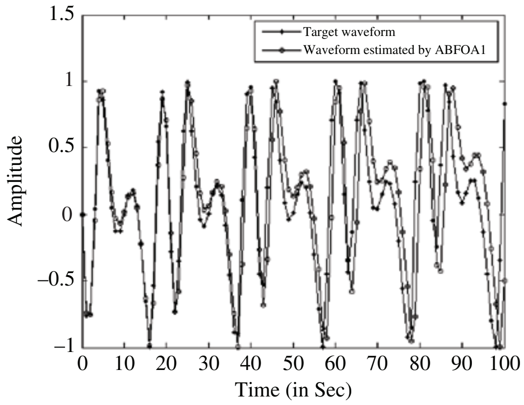


Fig. 16. Actual target sound and the waveform synthesized by ABFOA1.

the target sound and synthesized sounds are used to compute the fitness value. The process repeats until synthesized sounds become very similar to the target.

The specific instance of the problem discussed here involves determination of six real parameters $\vec{X} = \{a_1, \omega_1, a_2, \omega_2, a_3, \omega_3\}$ of the FM sound wave given by (32) for approximating it to the sound wave given in (34) where $\theta = 2\pi/100$. The parameters are defined in the range [−6.4, +6.35]

$$y(t) = a_1.\sin(\omega_1.t.\theta + a_2.\sin(\omega_2.t.\theta + a_3.\sin(\omega_3.t.\theta))) \quad (33)$$

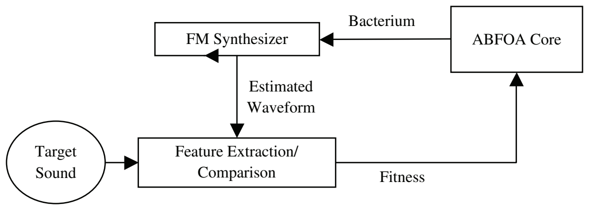


Fig. 17. Architecture of the optimization system.

$$y_0(t) = 1.0.\sin(5.0.t.\theta - 1.5.\sin(4.8.t.\theta + 2.0.\sin(4.9.t.\theta))). \quad (34)$$

The goal is to minimize the sum of square errors given by (35). This problem is a highly complex multimodal function having strong epistasis (interrelation among the variables), with the optimum value 0.0

$$f(\vec{X}) = \sum_{t=0}^{100} (y(t) - y_0(t))^2. \quad (35)$$

Due to the high difficulty of solving this problem with high accuracy without specific operators for continuous optimization (like gradual GAs [31]), we terminate the algorithm when either the error falls below 0.001 or the number of FEs exceed 10^6. Like the previous experiments, here also each run of the classical BFOA, ABFOA1, and ABFOA2 start with the same initial population.

In Table VII, we indicate the mean and the standard deviation (within parentheses) of the best-of-run values for 50 independent runs of each of the six algorithms over the FM synthesizer problem. The t-test performed on the data of Table VII indicates that the final mean accuracy of both the adaptive variants differ from their nearest competitor EA in a statistically significant fashion within 95% confidence interval. Table VIII shows, for all algorithms, the number of runs (out of 50) that managed to find the optimum at or below 0.001 without exceeding the maximum number of FEs. The table also reports the average number of FEs taken to reach the optima along with the standard deviation (in parentheses). Fig. 16 shows the convergence characteristics of six algorithms in terms of the objective function value versus number of FEs in their median run. Finally, in Fig. 17 we show

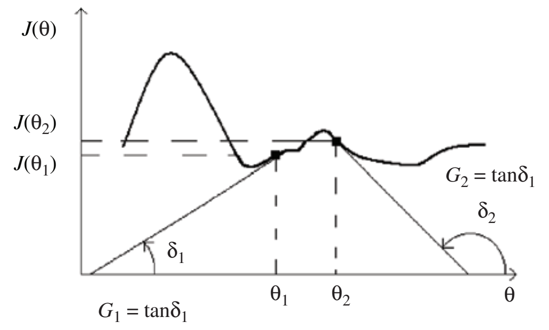


Fig. 18. Two-bacterium system on a 1-D fitness landscape.

the target waveform and synthesized waveform by ABFOA1, which yields the closest approximation of the target wave.

Tables VII and VIII and Fig. 16 indicate the superior performance of ABFOA1 over all its contestant algorithms in terms of final accuracy, convergence speed, and robustness. Fig. 17 shows that the waveform estimated by ABFOA1 achieves a high level of correspondence with the actual FM sound wave.

VIII. Conclusion

This paper has presented a simple mathematical analysis of the computational chemotaxis used in the BFOA. It also proposed simple schemes to adapt the chemotactic step size in BFOA with a view to improving its convergence behavior without imposing additional requirements in terms of the number of FEs. It has analytically been shown that the proposed adaptation schemes can avoid the oscillation around the optima or the stagnation near optima for a 1-D bacterium cell. The classical BFOA was compared with the adaptive BFOAs and a few other well-known evolutionary and swarm-based algorithms over a test bed of 10 well-known numerical benchmarks. The following performance metrics were used: 1) solution quality; 2) speed of convergence; and 3) frequency of hitting the optimum. The adaptive BFO variants were shown to provide better results than their classical counterpart for all of the tested problems. Moreover, the adaptive schemes outperformed a state-of-the-art variant of PSO, a standard real-coded GA, and a hybrid algorithm based on PSO and BFOA in a statistically meaningful fashion.

Although the adaptive schemes yielded superior results in the majority of the test cases, we must remember that this paper *does not* primarily aim at proposing a series of improved BFOA variants. Rather, it tries to understand how the chemotactic operator contributes to the search mechanism of BFOA, from a mathematical point of view. We believe that the performance of the competitor algorithms may also be enhanced with judicious parameter tuning, which renders itself to further research with them. However, the only conclusion we can draw at this point is that the adaptive chemotactic operators have an edge over the classical chemotaxis, especially in context to the convergence behavior of the algorithm very near the optima. This fact has been supported here both analytically and experimentally.

Future research may focus on extending the analysis presented in this paper to a group of bacteria working on a multidimensional fitness landscape and also include effect of the reproduction and elimination–dispersal events in the same. Other adaptation schemes of the chemotactic step size may also be investigated as well.

Appendix

Here, we are interested in the grouped behavior of two bacteria. Cost function value for each bacterium is modified if we consider mutual signaling between two bacteria. Let, θ_1 and θ_2 be the θ coordinates of two bacteria placed within the 1-D search space. $J(\theta_1)$ and $J(\theta_2)$ are the corresponding cost functions for their current positions as shown in Fig. 18. Let us also assume that V_1 and V_2 represent their velocities. After modifying cost function (J) by adding a cell-to-cell signaling function (J_{cc}), it becomes J' (say), where

$$\therefore J'(\theta_1) = J(\theta_1) + J_{cc} \text{ and } J'(\theta_2) = J(\theta_2) + J_{cc}$$

and

$$\begin{aligned} J_{cc} = &-d_a \exp\{-\omega_a(\theta_2-\theta_1)^2\} \\ &+ h_r \exp\{-\omega_r(\theta_2-\theta_1)^2\} \quad \text{[from (2)]} \end{aligned}$$

where ω_a and ω_r stand for $\omega_{\text{attractant}}$ and $\omega_{\text{repellant}}$, respectively, and d_a and h_r represent $d_{\text{attractant}}$ and $h_{\text{repellant}}$, respectively

$$\begin{aligned} \Rightarrow J_{cc} = &-h[\exp\{-\omega_a(\theta_2-\theta_1)^2\} \\ &- \exp\{-\omega_r(\theta_2-\theta_1)^2\}][\because d_a = h_r = h(\text{say})] \end{aligned}$$

From the above expressions, we can infer that

1) $J'(\theta_1)$, which is the modified cost function value for bacterium at θ_1, depends on θ_1 as well as on θ_2. In other words, the modified cost function for a particular bacterium may change even if it remains standstill, given that the other bacterium changes its position.

2) The cell-to-cell signal phenomenon can affect only the local search in BFOA. To illustrate this, let us assume $|\theta_2 - \theta_1| >> 1$, i.e., and a large distance separates two bacteria.

As $\exp\{-\omega_a(\theta_2-\theta_1)^2\} \to 0, \exp\{-\omega_r(\theta_2-\theta_1)^2\} \to 0$, we may have $J_{cc} \to 0$. It is clear that when the two bacteria are far away from each other, cell-to-cell signaling effect becomes feeble and the situation resembles that for a single bacterium system discussed earlier. Similar to one bacterium system, here we assume fitness landscape to be flat in the region of interest.

Analysis:

Similar to the one-bacterium system, the unit step function is used to model chemotaxis in this case

$$\Rightarrow \frac{\Delta\theta_1}{\Delta t} = u\left[-\frac{J'(\theta_1+\Delta\theta_1) - J'(\theta_1)}{\Delta t}\right].C.\Delta.$$

θ_1 and θ_2 are functions of t. When t changes, θ_1 and θ_2 change, causing $J'(\theta_1)$ to change. Thus when $\Delta t \to 0$, the above relation becomes

$$V_1 = \frac{d\theta_1}{dt} = u\left[-\frac{dJ'(\theta_1)}{dt}\right].C.\Delta. \tag{36}$$

$$V_1 = \frac{d\theta_1}{dt} = u\left[-\frac{dJ'(\theta_1)}{dt}\right].C.\Delta.$$

Similarly,

$$V_2 = \frac{d\theta_2}{dt} = u\left[-\frac{dJ'(\theta_2)}{dt}\right].C.\Delta. \tag{37}$$

Already we have shown that $J'(\theta_1)$ depends on both of θ_1 and θ_2. Again, θ_1 and θ_2 depend on time. Applying the rule of total derivatives, we get

$$\frac{dJ'(\theta_1)}{dt} = \frac{\partial J'(\theta_1)}{\partial\theta_1}\frac{d\theta_1}{dt} + \frac{\partial J'(\theta_1)}{\partial\theta_2}\frac{d\theta_2}{dt}. \tag{38}$$

Substituting the value of $J'(\theta_1)$ in (38), performing partial differentiation, and letting $\psi = 2h(\theta_2-\theta_1)[\omega_a \exp\{-\omega_a(\theta_2-\theta_1)^2\} - \omega_r \exp\{-\omega_r(\theta_2-\theta_1)^2\}]$, finally we get

$$\frac{dJ'(\theta_1)}{dt} = \left(\frac{dJ(\theta_1)}{d\theta_1} - \psi\right)\frac{d\theta_1}{dt} + \psi\frac{d\theta_2}{dt}$$

$$\Rightarrow \frac{dJ'(\theta_1)}{dt} = (G_1-\psi)V_1 + \psi V_2. \tag{39}$$

Similarly, we can show that

$$\frac{dJ'(\theta_2)}{dt} = (G_2+\psi)V_2 - \psi V_1 \tag{40}$$

where $G_1 = (dJ(\theta_1))/(d\theta_1)$ and $G_2 = (dJ(\theta_2))/(d\theta_2)$.

We assumed $|\theta_2 - \theta_1| << 1$, so neglecting higher order terms in expansion of e^{-x}, we get

$$\psi \approx 2h(\theta_2-\theta_1)(\omega_a-\omega_r)[1-(\omega_a+\omega_r)(\theta_2-\theta_1)^2]. \tag{41}$$

ψ, G_1, G_2, V_1, V_2 are small. Hence, $(dJ'(\theta_1))/(dt)$ and $(dJ'(\theta_2))/(dt)$ are small. So, unit step function in (36) and (37) can be modeled using the relation $u(x) \approx (k/4)x + (1/2)$

Thus,

$$V_1 = \frac{d\theta_1}{dt} = u\left[-\frac{dJ'(\theta_1)}{dt}\right]C\Delta \approx \left[-\frac{k}{4}\frac{dJ'(\theta_1)}{dt} + \frac{1}{2}\right] \times C\Delta \tag{42}$$

and

$$V_2 = \frac{d\theta_2}{dt} = u\left[-\frac{dJ'(\theta_2)}{dt}\right]C\Delta \approx \left[-\frac{k}{4}\frac{dJ'(\theta_2)}{dt} + \frac{1}{2}\right] \times C\Delta. \tag{43}$$

Substituting the values of $(dJ'(\theta_1))/(dt)$ and $(dJ'(\theta_2))/(dt)$ from (39) and (40) to (42) and (43), respectively

$$[4 + kC\Delta(G_1-\psi)]V_1 + kC\Delta\psi V_2 = 2C\Delta \tag{44}$$

and

$$[4 + kC\Delta(G_2+\psi)]V_2 - kC\Delta\psi V_1 = 2C\Delta. \tag{45}$$

The above equations show that velocities of two bacteria are coupled. We solve for V_1 from (44) and (45)

$$V_1 = \frac{8C\Delta + 2kC^2\Delta^2 G_2}{16 + k^2C^2\Delta^2\{G_1G_2 + \psi(G_1-G_2)\} + 4kC\Delta(G_1+G_2)}$$

for $\Delta^2 = 1[\because \Delta = 1 \text{or} \Delta = -1]$, and neglecting product of gradients

$$V_1 \approx \frac{\frac{C\Delta}{2} + \frac{kC^2}{8}G_2}{1 + \frac{k^2C^2}{16}\psi(G_1-G_2) + \frac{kC\Delta}{4}(G_1+G_2)}.$$

(Now, in the denominator 1 is much larger than remaining terms. So, we use approximate relation, $(1)/(1+x) \approx 1-x$ for simplification.)

$$V_1 \approx \frac{C\Delta}{2} - \frac{kC^2}{8}G_1 - \frac{k\Delta}{32}[k\psi(G_1-G_2) + G_2(G_1+G_2)]C^3 - \frac{k^3\psi}{128}G_2(G_1-G_2)C^4. \tag{46}$$

It is general equation for the two-bacteria system. A careful inspection of (45) reveals that

1) $V_1 = a_0C + a_1C^2 + a_2C^3 + a_3C^4$ where, $a_i = f(G_1, G_2, \theta_2, \theta_1, \Delta, \omega_a, \omega_r, h, k)$, i.e., the velocity can be expressed as a polynomial of step size.

2) C is very small. So, last two terms on the RHS of (46) have lesser contributions. We conclude that first two terms dominate the relation, i.e., $V_1 \approx (C\Delta/2) - ((kC^2)/8)G_1$. This is the equation obtained for single-bacterium system.

3) When $k\psi(G_1-G_2) >> G_2(G_1+G_2)$, we neglect latter as a coefficient of C^3.

Also, substituting $\psi = 2h(\theta_2-\theta_1)[\omega_a \exp\{-\omega_a(\theta_2-\theta_1)^2\} - \omega_r \exp\{-\omega_r(\theta_2-\theta_1)^2\}]$, we get

$$V_1 \approx -\frac{kC^2}{8}G_1 + \frac{C.\Delta}{2} + \tau(\theta_2-\theta_1) \tag{47}$$

where $\tau = f(G_1, G_2, C, \theta_2, \theta_1, \Delta, \omega_a, \omega_r, h, k)$. It has a very complicated form, but its magnitude is small. Expression (46) reveals an interesting feature of the two-bacteria system. The first two terms on the RHS are exactly similar to those of the one-bacterium system described by (12). But the third term, i.e., $\tau(\theta_2-\theta_1)$, shows swarming effect. This term is introduced due to the effect of other bacterium in its dynamics, although its magnitude is considerably smaller than the first two terms in (47).

References

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

[2] Y. Liu and K. M. Passino, "Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors," *J. Optimization Theory Applicat.*, vol. 115, no. 3, pp. 603–628, Dec. 2002.

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

[4] S. Mishra, "A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 61–73, Feb. 2005.

[5] M. Tripathy, S. Mishra, L. L. Lai, and Q. P. Zhang, "Transmission loss reduction based on FACTS and bacteria foraging algorithm," in *Proc. PPSN*, 2006, pp. 222–231.

[6] S. Mishra, and C. N. Bhende, "Bacterial foraging technique-based optimized active power filter for load compensation," *IEEE Trans. Power Delivery*, vol. 22, no. 1, pp. 457–465, Jan. 2007.

[7] D. H. Kim and C. H. Cho, "Bacterial foraging based neural network fuzzy learning," in *Proc. IICAI 2005*, pp. 2030–2036.

[8] B. D. Hughes, *Random Walks and Random Environments: Volume 1: Random Walks*. London, U.K.: Oxford Univ. Press, 1996.

[9] M. Avriel, *Nonlinear Programming: Analysis and Methods*. 1st ed. New York: Dover, 2003. ch. 10, sec. 10.1, pp. 288–299.

[10] J. A. Snyman, *Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*. New York: Springer-Verlag, 2005.

[11] C. Darken, and J. Moody, "Toward faster stochastic gradient search," in *Proc. Advances Neural Inform. Process. Syst. (NIPS) 4*, Palo Alto, CA: Morgan Kaufmann, 1992, pp. 1009–1016.

[12] L. Armijo, "Minimization of functions having Lipschitz–continuous first partial derivatives," *Pacific J. Math.*, vol. 16, no. 1, pp. 1–3, 1966.

[13] V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis, "Learning rate adaptation in stochastic gradient descent," in *Proc. Advances Convex Anal. Global Optimization, Noncovex Optimization Applicat.*, vol. 54, Dordrecht, Netherlands: Kluwer, 2001, ch. 27, pp. 433–444.

[14] H. C. Hsin, C. C. Li, M. Sun, and R. J. Sclabassi, "An adaptive training algorithm for back–propagation neural networks," *IEEE Trans. Syst., Man, Cybern.*, vol. 25, no. 3, pp. 512–514, Mar. 1995.

[15] G. B. Orr and T. K. Leen, "Using curvature information for fast stochastic search," in *Proc. Advances Neural Inform. Process. Syst. 9*, Cambridge, MA: MIT Press, 1997.

[16] M. Giudici, M. Queirolo, and M. Valle, "Evaluation of gradient descent learning algorithms with adaptive and local learning rate for recognizing hand-written numerals," in *Proc. ESANN*, Bruges, Belgium, 2002, pp. 289–294.

[17] Y. Le Cun, P. Y. Simard, and B. A. Pearlmutter, "Automatic learning rate maximization by on–line estimation of the Hessian's eigenvectors," in *Proc. Eur. Symp. Artificial Neural Netw. (ESANN 2002)*, pp. 156–163.

[18] D. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, 1963.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Harbor, MI: Univ. Michigan Press, 1975.

[21] T. Bäck, "Evolutionary algorithms in theory and practice: Evolution strategies, evol. programming," in *Proc. Genetic Algorithms*, London, U.K.: Oxford Univ. Press, 1996.

[22] E. Budrene and H. Berg, "Dynamics of formation of symmetrical patterns by chemotactic bacteria," *Nature*, vol. 376, pp. 49–53, Jul. 1995.

[23] Y. Blat and M. Eisenbach, "Tar-dependent and independent pattern formation by salmonella typhimurium," *J. Bacteriology*, vol. 177, no. 7, pp. 1683–1691, Apr. 1995.

[24] W. J. Tang, Q. H. Wu, and J. R. Saunders, "A novel model for bacteria foraging in varying environments," in *Proc. ICCSA*, LNCS vol. 3980. 2006, pp. 556–565.

[25] M. S. Li, W. J. Tang, W. H. Tang, Q. H. Wu, and J. R. Saunders, "Bacteria foraging algorithm with varying population for optimal power flow," in *Proc. Evol. Workshops 2007*, LNCS vol. 4448. pp. 32–41.

[26] M. Tripathy and S. Mishra, "Bacteria foraging-based to optimize both real power loss and voltage stability limit," *IEEE Trans. Power Syst.*, vol. 22, no. 1, pp. 240–248, 2007.

[27] A. Biswas, S. Dasgupta, S. Das, and A. Abraham, "Synergy of PSO and bacterial foraging optimization: A comparative study on numerical benchmarks," in *Proc. 2nd Int Symp. Hybrid Artificial Intell. Syst. (HAIS) Advances Soft Computing Ser.*, vol. 44. Germany, Springer-Verlag, Innovations in Hybrid Intelligent Systems, ASC, 2007, pp. 255–263.

[28] L. Ulagammai, P. Vankatesh, P. S. Kannan, and N. P. Padhy, "Application of bacteria foraging technique trained and artificial and wavelet neural networks in load forecasting," *Neurocomputing*, vol. 70, no. 16–18, pp. 2659–2667, Oct. 2007.

[29] M. A. Munoz, J. A. Lopez, and E. Caicedo, "Bacteria foraging optimization for dynamical resource allocation in a multizone temperature experimentation platform," *Anal. Des. Intel. Sys. Using SC Tech, ASC*, vol. 41, pp. 427–435, Sep. 2007.

[30] D. P. Acharya, G. Panda, S. Mishra, and Y. V. S. Lakhshmi, "Bacteria foaging based independent component analysis," in *Proc. Int. Conf. Comput. Intell. Multimedia Applicat.* Piscataway, NJ: IEEE Press, 2007, pp. 527–531.

[31] A. Chatterjee and F. Matsuno, "Bacterial foraging techniques for solving EKFBased SLAM problems," in *Proc. Int. Control Conf.*, Glasgow, U.K., Aug. 30–Sep. 1, 2006.

[32] R. P. Anwal, *Generalized Functions: Theory and Technique*. 2nd ed. Boston, MA: Birkhãuser, 1998.

[33] R. Fletcher, *Practical Methods of Optimization*. 2nd ed. Chichester, U.K.: Wiley, 1987.

[34] K. Price, R. Storn, and J. Lampinen, *Differential Evolution A Practical Approach to Global Optimization*. Berlin, Germany: Springer-Verlag, 2005.

[35] H.-G. Beyer, "Theory of Evolution Strategies," in *Proc. Natural Computing Ser.*. Berlin, Germany, Springer-Verlag, 2001.

[36] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. New York: Wiley, 1966.

[37] H.-G. Beyer, "Toward a theory of evolution strategies: Self-adaptation," *Evol. Comput.*, vol. 3, no. 3, pp. 311–347, 1996.

[38] H. Kita, "A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms," *Evol. Comput.*, vol. 9, no. 2, pp. 223–241, 2001.

[39] N. Saravanan and D. B. Fogel, "Learning of strategy parameters in evolutionary programming: An empirical study," in *Proc. 3rd Annu. Conf. Evol. Programming*, River Edge, NJ: World Scientific, 1994, pp. 269–280.

[40] D. B. Fogel, L. J. Fogel, and W. Atmar, "Meta-evolutionary programming," in *Proc. 25th Asilomar Conf. Signals, Syst. Computers*, Pacific Grove, CA, 1991, pp. 540–545.

[41] D. B. Fogel, "Evolving Artificial Intelligence," Ph.D. dissertation, Dept. Eng., Univ. California, San Diego, CA, 1992.

[42] C. Y. Lee and X. Yao, "Evolutionary programming using the mutations based on the Lévy probability distribution," *IEEE Trans. Evol. Comput.*, vol. 8, no. 1, pp. 1–13, Jan. 2004.

[43] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 82–102, 1999.

[44] P. J. Angeline, "Evolutionary optimization versus particle swarm optimization: Philosophy and the performance difference," in *Proc. 7th Int. Conf. Evol. Programming Evol. Programming*, LNCS vol. 1447. 1998, pp. 84–89.

[45] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[46] A. Ratnaweera and K. S. Halgamuge, "Self organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–254, 2004.

[47] D. Ashlock, *Evolutionary Computation for Modeling and Optimization*. New York: Springer-Verlag, 2006.

[48] R. Thomsen, "Flexible ligand docking using evolutionary algorithms: Investigating the effects of variation operators and local search hybrids," *BioSyst.*, vol. 72, no. 1–2, pp. 57–73, 2003.

[49] B. Flury, *A First Course in Multivariate Statistics*. vol. 28, New York: Springer-Verlag, 1997.

[50] A. Horner, J. Beauchamp, and L. Haken, "Genetic algorithms and their application to FM matching synthesis," *Comput. Music, J.*, vol. 17, pp. 17–29, 1993.

[51] F. Herrera and M. Lozano, "Gradual distributed real-coded genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 4, no. 1, pp. 43–62, Apr. 2000.

Sambarta Dasgupta was born in Kolkata, India, in 1987. He obtained the B.E.Tel.E degree from the Department of Electronics and Telecommunication Engineering, Jadavpur University, India in 2009.

His research interests include the theory of evolutionary computing, swarm intelligence, image processing, robotics, and control engineering. He has published around 20 research articles in refereed international journals and reputed international conferences, including GECCO-2008 and CEC-2008.

Swagatam Das was born in Kolkata, India, in 1980. He received the B.E.Tel.E. and the M.E.Tel.E. degrees, both in control engineering specialization, and the Ph.D. degree in engineering, from Jadavpur University, India, in 2003, 2005, and 2009 respectively.

He is presently serving as a Lecturer in the Department of Electronics and Telecommunication Engineering, Jadavpur University. His current research interests include evolutionary computing, swarm intelligence, pattern recognition, bioinformatics, control systems engineering, and digital signal processing. He has published more than 70 research articles in peer-reviewed journals and international conferences. He coauthored a monograph on metaheuristics-based clustering published by Springer in April, 2009. He has been acting as a reviewer for journals like *Pattern Recognition*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, PART B, IEEE/ACM TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, *Elsevier Signal Processing, Information Sciences, and Neurocomputing*.

Dr. Das is the recipient of the best paper award in the sixth International Conference on Intelligent Systems Design and Applications (ISDA2006), held in Jinan, Shandong, China. He serves as the editorial board member of the *International Journal of Artificial Intelligence and Soft Computing* (IJAISC), Inderscience Publications.

Ajith Abraham (M'96–SM'07) received the Ph.D. degree from Monash University, Melbourne, Australia, in 2001.

He is currently with the Center for Quantifiable Quality of Service in Communication Systems, Norwegian University of Science and Technology (NTNU), Trondheim, Norway, and also with the Machine Intelligence Research Labs (MIR Labs). He has worldwide academic experience of nearly ten years, with formal appointments in Monash University, Australia; Oklahoma State University, Stillwater; Chung-Ang University, Seoul; Jinan University, China; Rovira i Virgili University, Spain; Dalian Maritime University, China; Yonsei University, Seoul; Open University of Catalonia, Spain; and NTNU, Norway. For about two and a half years, he was working under the Institute of Information Technology Advancement Professorship Program funded by the South Korean Government. He also was with three multinational companies: Keppel Engineering, Singapore; Hyundai Engineering, Korea; and Ashok Leyland Ltd, India, where he was involved in different industrial research and development projects for nearly eight years. He has authored or coauthored more than 400 research publications in peer-reviewed reputed journals, book chapters, and conference proceedings. His primary research interests are in advanced computational intelligence, with a focus on using global optimization techniques for designing intelligent systems. Application areas include Web services, information security, Web intelligence, social networks, financial modeling, multicriteria decision making, data mining, etc. He has given more than 25 plenary lectures and conference tutorials in these areas.

Dr. Abraham co-chairs the IEEE Systems, Man, and Cybernetics Society Technical Committee on Soft Computing. He is a regular reviewer of IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON NEURAL NETWORKS, IEEE TRANSACTIONS ON FUZZY SYSTEMS, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, and IEEE TRANSACTIONS ON POWER SYSTEMS. He serves on the Editorial Boards of more than 30 international journals and has also guest-edited 30 special issues on various topics for international journals. Since 2001, he has been actively involved in the hybrid intelligent systems and the Intelligent Systems Design and Applications series of annual international conferences. He was the recipient of five Best Paper Awards.

Arijit Biswas was born in Kolkata, India, in 1987. He obtained the B.E.Tel.E. degree from the Electronics and Telecommunication Engineering Department, Jadavpur University, India in 2009.

His research interests include artificial intelligence, optimization, computer vision, and machine learning. He has published twenty research articles in refereed international journals and proceedings of reputed international conferences, including GECCO-2008 and CEC-2008.