Available online at www.sciencedirect.com
ScienceDirect
Information Sciences 177 (2007) 3918–3937
INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL
www.elsevier.com/locate/ins

A hybrid genetic algorithm and bacterial foraging approach for global optimization

Dong Hwa Kim [a], Ajith Abraham [b,*], Jae Hoon Cho [a]

[a] *Department of Instrumentation and Control Engineering, Hanbat National University, 16-1 San Duckmyong-Dong Yuseong-Gu, Daejon 305-719, Republic of Korea*

[b] *Center of Excellence for Quantifiable Quality of Service (Q2S), Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, O.S. Bragstads plass 2E, N-7491 Trondheim, Norway*

Received 14 June 2005; received in revised form 22 March 2007; accepted 2 April 2007

Abstract

The social foraging behavior of *Escherichia coli* bacteria has been used to solve optimization problems. This paper proposes a hybrid approach involving genetic algorithms (GA) and bacterial foraging (BF) algorithms for function optimization problems. We first illustrate the proposed method using four test functions and the performance of the algorithm is studied with an emphasis on mutation, crossover, variation of step sizes, chemotactic steps, and the lifetime of the bacteria. The proposed algorithm is then used to tune a PID controller of an automatic voltage regulator (AVR). Simulation results clearly illustrate that the proposed approach is very efficient and could easily be extended for other global optimization problems.

Keywords: Genetic algorithm; Bacterial foraging optimization; Hybrid optimization; Controller tuning

1. Introduction

In the last decade, approaches based on genetic algorithms (GA) have received increased attention from the academic and industrial communities for dealing with optimization problems that have been shown to be intractable using conventional problem solving techniques [10,13,14,20,25,31,32,34]. A typical task of a GA is to find the best values of a predefined set of free parameters associated with either a process model or a control vector. System identification is one of the active areas of GA research [3,4,33,40,41]. Recent surveys of genetic algorithms, relating to improvements in the search process with respect to control system engineering problems, can be found in [33,41,42]. GA has also been used extensively to optimize nonlinear systems. A large

* Corresponding author.
E-mail address: ajith.abraham@ieee.org (A. Abraham).

0020-0255/$ - see front matter
doi:10.1016/j.ins.2007.04.002

volume of research is focused on the design of fuzzy controllers using evolutionary algorithm approaches. GA is often used to develop the *if–then* linguistic knowledge base of the controlled process, and to fine tune the fuzzy membership function parameters [1].

Usually, a possible solution to a specific problem is encoded as an individual (or a chromosome), which consists of a group of genes. Each individual represents a point in the search space and a possible solution to the problem can be formulated. A population consists of a finite number of individuals and each individual is decided by a fitness evaluation. Using this fitness value and suitable genetic operators, a new population is generated iteratively, with each iteration referred to as a *generation*. The GA uses basic genetic operators such as crossover and mutation to produce the genetic composition of a population. The crossover operator produces two offspring (new candidate solutions) by recombining the information from two parents. As the mutation operation is a random alteration of some gene values in an individual, the allele of each gene is a candidate for mutation, and its applicability is determined by the mutation probability. In the literature, much research has gone into the enhancement of conventional genetic algorithms [2,5,21].

In the past, some researchers have focused on using hybrid genetic algorithm approaches for optimization problems. Buczak and Uhrig [7] proposed a novel hierarchical fuzzy-genetic information fusion technique. The combined reasoning takes place by means of fuzzy aggregation functions, capable of combining information by compensatory connectives that better mimic the human reasoning process than union and intersection, employed in traditional set theories. The parameters of the connectives are found by genetic algorithms.

Gómez-Skarmeta et al. [19] evaluated the use of different methods from the fuzzy modeling field for classification tasks and the potential of their integration in producing better classification results. The methods considered, approximate in nature, study the integration of techniques with an initial rule generation step and a following rule tuning approach using different evolutionary algorithms.

In order to discover classification rules, Carvalho and Freitas [8] proposed a hybrid decision tree/genetic algorithm method. The central idea of this hybrid method involves the concept of small disjunctions in data mining. The authors developed two genetic algorithms specifically designed for discovering rules in examples belonging to small disjunctions, whereas a conventional decision tree algorithm is used to produce rules covering examples belonging to large disjunctions. Lee and Lee [30] proposed a hybrid search algorithm combining the advantages of genetic algorithms and ant colony optimization (ACO) that can explore the search space and exploit the best solutions.

Constraint handling is one of the major concerns when applying genetic algorithms to solve constrained optimization problems. Chootinan and Chen [9] proposed gradient information, derived from the constraint set, to systematically repair infeasible solutions. The proposed repair procedure is embedded in a simple GA as a special operator. Haouari and Siala [22] presented a lower bound and a genetic algorithm for the prize collecting Steiner tree problem. The lower bound is based on a Lagrangian decomposition of a minimum spanning tree formulation of the problem.

Natural selection tends to eliminate animals with poor foraging strategies through methods for locating, handling, and ingesting food and favors the propagation of genes of those animals that have successful foraging strategies, since they are more likely to obtain reproductive success [36,37,12,35]. After many generations, poor foraging strategies are either eliminated or re-structured into good ones. Since a foraging organism/animal takes actions to maximize the energy utilized per unit time spent foraging, considering all the constraints presented by its own physiology, such as sensing and cognitive capabilities and environmental parameters (e.g., density of prey, risks from predators. physical characteristics of the search area), natural evolution could lead to optimization. It is essentially this idea that could be applied to complex optimization problems. The optimization problem search space could be modeled as a social foraging environment where groups of parameters communicate cooperatively for finding solutions to difficult engineering problems [27].

The rest of the paper is organized as follows. Section 2 provides a brief literature overview of the bacterial foraging algorithm followed by the proposed hybrid approach based on BF (Bacterial Foraging) and GA (genetic algorithms). The proposed algorithm is validated using four test functions and for PID controller tuning [18,29,17] in Section 3. Some conclusions are also provided towards the end.

2. Hybrid system consisting of genetic algorithm and bacteria foraging

2.1. Genetic algorithms

In nature, evolution is mostly determined by natural selection, where individuals that are better are more likely to survive and propagate their genetic material. The encoding of genetic information (genome) is done in a way that admits asexual reproduction which results in offspring's that are genetically identical to the parent. Sexual reproduction allows some exchange and re-ordering of chromosomes, producing offspring that contain a combination of information from each parent. This is the recombination operation, which is often referred to as crossover because of the way strands of chromosomes crossover during the exchange. Diversity in the population is achieved by *mutation*. A typical genetic algorithm procedure takes the following steps: A population of candidate solutions (for the optimization task to be solved) is initialized. New solutions are created by applying genetic operators (mutation and/or crossover). The fitness (how good the solutions are) of the resulting solutions are evaluated and suitable selection strategy is then applied to determine which solutions will be maintained into the next generation. The procedure is then iterated.

Genetic algorithms are ubiquitous nowadays, having been successfully applied to numerous problems from different domains, including optimization, automatic programming, machine learning, operations research, bioinformatics, and social systems.

2.2. Bacterial foraging algorithm

Recently, search and optimal foraging of bacteria have been used for solving optimization problems [16]. To perform social foraging, an animal needs communication capabilities and over a period of time it gains advantages that can exploit the sensing capabilities of the group. This helps the group to predate on a larger prey, or alternatively, individuals could obtain better protection from predators while in a group.

2.2.1. Overview of chemotactic behavior of Escherichia coli

In our research, we considered the foraging behavior of *E. coli*, which is a common type of bacteria [36,37]. Its behavior and movement comes from a set of six rigid spinning (100–200 r.p.s) flagella, each driven as a biological motor. An *E. coli* bacterium alternates through running and tumbling. Running speed is 10–20 μm/s, but they cannot swim straight. The chemotactic actions of the bacteria are modeled as follows:

- In a neutral medium, if the bacterium alternatively tumbles and runs, its action could be similar to search.
- If swimming up a nutrient gradient (or out of noxious substances) or if the bacterium swims longer (climb up nutrient gradient or down noxious gradient), its behavior seeks increasingly favorable environments.
- If swimming down a nutrient gradient (or up noxious substance gradient), then search action is like avoiding unfavorable environments.

Therefore, it follows that the bacterium can climb up nutrient hills and at the same time avoids noxious substances. The sensors it needs for optimal resolution are receptor proteins which are very sensitive and possess high gain. That is, a small change in the concentration of nutrients can cause a significant change in behavior. This is probably the best-understood sensory and decision-making system in biology [16].

Mutations in *E. coli* affect the reproductive efficiency at different temperatures, and occur at a rate of about 10^{-7} per gene per generation. *E. coli* occasionally engages in a conjugation that affects the characteristics of the population. There are many types of taxis that are used in bacteria such as, aerotaxis (attracted to oxygen), phototaxis (light), thermotaxis (temperature), magnetotaxis (magnetic lines of flux) and some bacteria can change their shape and number of flagella (based on the medium) to reconfigure in order to ensure efficient foraging in a variety of media. Bacteria could form intricate stable spatio-temporal patterns in certain semi-solid nutrient substances and they can survive through a medium if placed together initially at its center. Moreover, under certain conditions, they will secrete cell-to-cell attractant signals so that they will group and protect each other.

2.2.2. The optimization function for the hybrid genetic algorithm–bacterial foraging (GA–BF) algorithm

The main goal of the Hybrid GA–BF based algorithm is to find the minimum of a function $P(\phi), \phi \in R^n$, which is not in the gradient $\nabla P(\phi)$. Here, ϕ is the position of a bacterium, and $P(\phi)$ is an attractant–repellant profile. That is, where nutrients and noxious substances are located, $P < 0$, $P = 0$ and $P > 0$ represents the presence of nutrients. A neutral medium, and the presence of noxious substances, respectively can be defined by

$$H(j,k,l) = \{\phi^x(j,k,l)|x = 1,2,\ldots,N\}. \tag{1}$$

Eq. (1) represents the position of each member in the population of N bacteria at the jth chemotactic step, kth reproduction step, and lth elimination–dispersal event. Let $P(x,j,k,l)$ denote the cost at the location of the ith bacterium at position

$$\phi^x(i,j,k) \in R^n \quad \text{and} \quad \phi^x = (i+1,j,k) = \phi^x(i,j,k) + C(x)\varphi(i), \tag{2}$$

so that $C(i) > 0$ is the step size taken in the random direction specified by the tumble. If at $\phi^x(i+1,j,k)$ the cost $P(i,j+1,k,l)$ is better (lower) than at $\phi^x(i,j,k)$, then another chemotactic step of size $C(x)$ in this same direction will be taken and repeated up to a maximum number of N_s steps. N_s is the length of the lifetime of the bacteria measured by the number of chemotactic steps. Function $P_c^i(\phi)$, $i = 1,2,\ldots,S$, to model the cell-to-cell signaling via an attractant and a repellant is represented by [23,39,36,37]

$$\begin{aligned} P_c(\phi) &= \sum_{i=1}^{N} P_{cc}^i \\ &= \sum_{i=1}^{N}\left[-L_{\text{attract}}\exp\left(-\delta_{\text{attract}}\sum_{j=1}^{n}(\phi_j - \phi_j^i)^2\right)\right] \\ &\quad + \sum_{i=1}^{N}\left[-K_{\text{repellant}}\exp\left(-\delta_{\text{attract}}\sum_{j=1}^{n}(\phi_j - \phi_j^i)^2\right)\right], \end{aligned} \tag{3}$$

where $\phi = [\phi_1,\ldots,\phi_p]^{\mathrm{T}}$ is a point on the search space, L_{attract} is the depth of the attractant released by the cell and δ_{attract} is a measure of the width of the attractant signal. $K_{\text{repellant}} = L_{\text{attract}}$ is the height of the repellant effect magnitude, and δ_{attract} is a measure of the width of the repellant. The expression $P_c(\phi)$ means that its value does not depend on the nutrient concentration at position ϕ. That is, a bacterium with high nutrient concentration secretes stronger attractant than one with low nutrient concentration. The model uses the function $P_{ar}(\phi)$ to represent the environment-dependent cell-to-cell signaling as

$$P_{ar}(\phi) = \exp(T - P(\phi))P_c(\phi), \tag{3a}$$

where T is a tunable parameter. By considering the minimization of $P(i,j,k,l) + P_{ar}(\phi^i(j,k,l))$, the cells try to find nutrients, avoid noxious substances, and at the same time try to move toward other cells, but not too close to them. The function $P_{ar}(\phi^i(j,k,l))$ implies that, with T being constant, the smaller the value of $P(\phi)$, the larger $P_{ar}(\phi)$ and thus the stronger the attraction, which is intuitively reasonable. For tuning the parameter T, it is normally found that, when T is very large, $P_{ar}(\phi)$ is much larger than $J(\phi)$, and thus the profile of the search space is dominated by the chemical attractant secreted by *E. coli*. On the other hand, if T is very small, then $P_{ar}(\phi)$ is much smaller than $P(\phi)$, and it is the effect of the nutrients that dominates. In $P_{ar}(\phi)$, the scaling factor of $P_c(\phi)$ is given as in exponential form.

The algorithm to search optimal values of parameters is described as follows:

[**Step 1**] Initialize parameters $n, N, N_C, N_s, N_{re}, N_{ed}, P_{ed}$, $C(i)(i = 1,2,\ldots,N)$, ϕ^i.
where,
n: Dimension of the search space,
N: The number of bacteria in the population,
N_C: Chemotactic steps,
N_{re}: The number of reproduction steps,
N_{ed}: The number of elimination–dispersal events,

P_{ed}: Elimination–dispersal with probability,

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

[**Step 2**] Elimination–dispersal loop: $l = l+1$.

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

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

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

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

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

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

[substep e] Move: Let

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

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

[substep f] Compute $\mathrm{ITSE}(i,j+1,k,l)$.

[substep g] Swim.

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

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

- Let $m = m+1$.
- If $\mathrm{ITSE}(i,j+1,k,l) < \mathrm{ITSE}_{\mathrm{last}}$(if doing better), let $\mathrm{ITSE}_{\mathrm{last}} = \mathrm{ITSE}\ (i,j+1,k,l)$ and let

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

and use this $\phi^x(i+1,j,k)$ to compute the new $\mathrm{ITSE}(i,j+1,k,l)$ as we did in [substep f].

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

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

[**Step 5**] If $j < N_C$, go to step 3. In this case, continue chemotaxis, since the life of the bacteria is not over.

[**Step 6**] Reproduction:

[substep a] For the given k and l, and for each $i = 1,2,\ldots,N$, let

$$\mathrm{ITSE}^i_{\mathrm{health}} = \sum_{j=1}^{N_c+1} \mathrm{ITSE}(i,j,k,l)$$

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 $\mathrm{ITSE}_{\mathrm{health}}$ (higher cost means lower health).

[substep b] The S_r bacteria with the highest $\mathrm{ITSE}_{\mathrm{health}}$ values die and the remaining S_r bacteria with the best values split (this process is performed by the copies that are made are placed 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 $i = 1,2,\ldots,N$, with probability P_{ed}, eliminate and disperse each bacterium, which results in keeping the number of bacteria in the population constant. To do this, if a bacterium is eliminated, simply disperse one to a random location on the optimization domain. If $l < N_{ed}$, then go to [step 2]; otherwise end.

3. Experiment results using test functions

This section illustrates some comparisons between the proposed GA–BF (genetic algorithm–Bacteria Foraging algorithm) and the conventional SGA (simple genetic algorithm) using some test functions as depicted in Table 1. Table 1 also illustrates the initial conditions of objective values, parameter values, Chemotactic Steps (CS), total number of chemotactic reaction of bacteria, step sizes, basic unit for movement of bacteria the number of critical reaction (N), the number of bacteria (S), generations (G), mutation (Mu), and crossover (Cr).

3.1. Mutation operation in GA–BF

Dynamic mutation [33] is used in the proposed GA–BF algorithm

$$x_j = \begin{cases} \tilde{x}_j + \Delta(k, x_j^{(U)} - \tilde{x}_j), & \tau = 0, \\ \tilde{x}_j - \Delta(k, \tilde{x}_j - x_j^{(L)}), & \tau = 1, \end{cases} \tag{4}$$

where the random constant τ becomes 0 or 1 and $\Delta(k, y)$ is given as

$$\Delta(k, y) = y \cdot \eta \cdot \left(1 - \frac{k}{z}\right)^A. \tag{5}$$

Here, $\eta = 0$ or 1 randomly and z is the maximum number of generations as defined by the user.

3.2. Crossover operation in GA–BF

A modified simple crossover [34] is used for the BF–GA algorithm using

$$\tilde{x}_j^u = \lambda \bar{x}_j^v + (1 - \lambda)\bar{x}_j^u, \tag{6a}$$

$$\tilde{x}_j^v = \lambda \bar{x}_j^u + (1 - \lambda)\bar{x}_j^v, \tag{6b}$$

where $\bar{x}_j^u, \bar{x}_j^v$ refers to parent's generations and $\tilde{x}_j^u, \tilde{x}_j^v$ refers to offspring's generations and j is the chromosome of jth step and λ is the multiplier.

3.3. Performance variation for different step sizes

Step size here refers to the moving distance per step of the bacteria. For performance comparison the following test function (F) is used as depicted in Fig. 1:

$$F(x) = \sum_{i=1}^{3} x_i^2, \quad -5.12 \leqslant x_1, x_2, x_3 \leqslant 5.11. \tag{7}$$

Figs. 2a, 2b and Table 2 illustrate the performance of the GA–BF algorithm for 300 generations. As evident from the results for bigger step size, the convergence is faster. Table 1 illustrates the empirical performance.

Table 1
Initial conditions for test functions and variation of different parameters

Test function	Range		Genetic algorithm parameters				Bacteria foraging parameters			
	$x_i^{(L)}$	$x_i^{(U)}$		G	Mu	Cr	CS	Step size	N_s	S
$F_1(x) = \sum_{i=1}^{3} x_i^2$	−5.12	5.11	20	300	0.9	0.1	1000	1e−007	3	10
$F_2(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$	−2.048	2.047	20	600	0.9	0.1	1000	1e−007	3	10
$F_3 = \sum_{i=1}^{5} [x_i]$	−5.12	5.12	20	180	0.9	0.1	1000	1e−007	3	10
$F_4 = \sum_{i=1}^{30} i x_i^4 + N(0, 1)$	−1.28	1.27	20	300	0.9	0.1	1000	1e−007	3	10

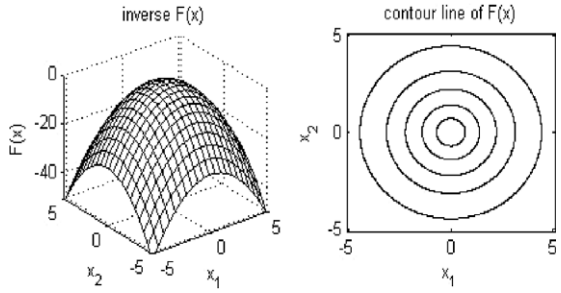

Fig. 1. Contour of test function F_1.

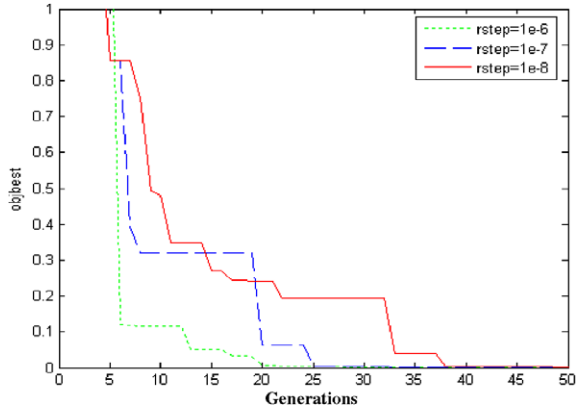

Fig. 2a. Performance value for the three different step sizes for the first 50 generations.

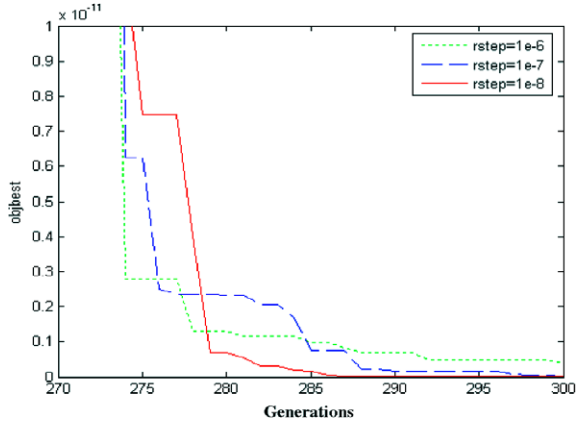

Fig. 2b. Performance value for the three different step sizes for generations 270–300.

Table 2
Parameter values for various step sizes

Step size	x_1	x_2	x_3	Optimal objective function	Average objective function
1.0e−6	3.87E−13	6.60E−13	2.92E−07	−5.43E−07	−8.98E−08
1.0e−7	2.85E−14	2.34E−13	−5.52E−08	1.50E−07	−5.45E−08
1.0e−8	5.01E−16	1.43E−15	−1.70E−08	−1.44E−08	−2.31E−09

3.4. Performance for different chemotactic steps of GA–BF

Figs. 2a, 2b and 3 and Table 3 illustrate the relationship between the objective function and the number of generations for different chemotactic steps. As evident, when the chemotactic step is smaller, the objective function converges faster.

3.5. Performance for different life time N_s

Figs. 4a and 4b illustrate the characteristics between objective function and the number of generations for different life time N_s of the bacteria.

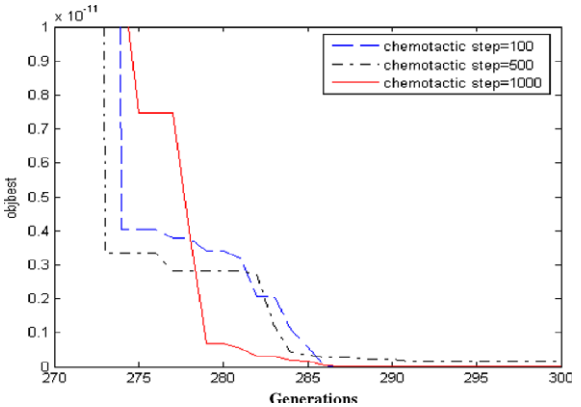

Fig. 3. Performance value for different chemotactic step for generations 270–300.

Table 3
Variation of objective function for different chemotactic steps

Chemotactic step size	x_1	x_2	x_3	Optimal objective function value	Average objective function value
100	−9.32E−08	3.78E−07	−8.57E−09	1.52E−13	1.59E−13
500	2.97E−08	1.92E−08	2.32E−08	1.79E−15	3.26E−15
1000	−1.70E−08	−1.44E−08	−2.31E−09	5.01E−16	1.43E−15

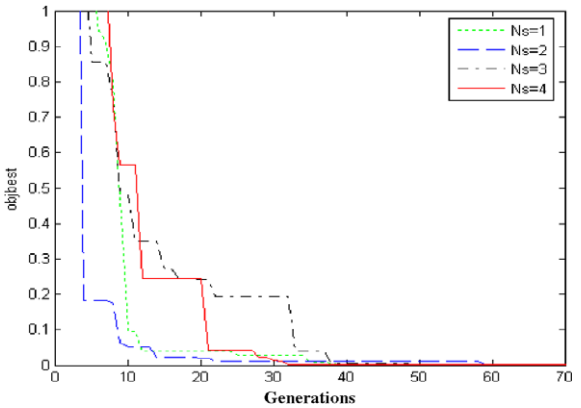

Fig. 4a. Performance value for different lifetime N_s for the first 70 generations.

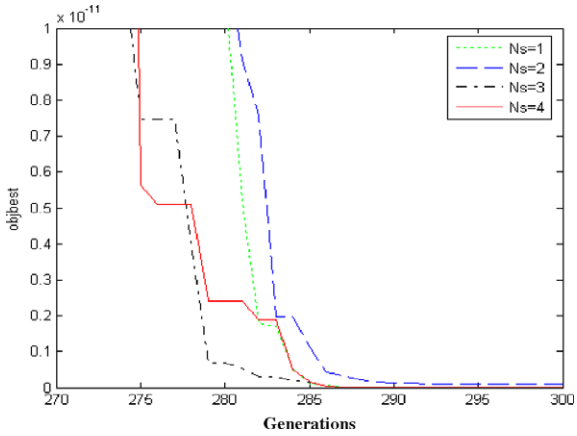

Fig. 4b. Performance value for different lifetime N_s for generations 270–300.

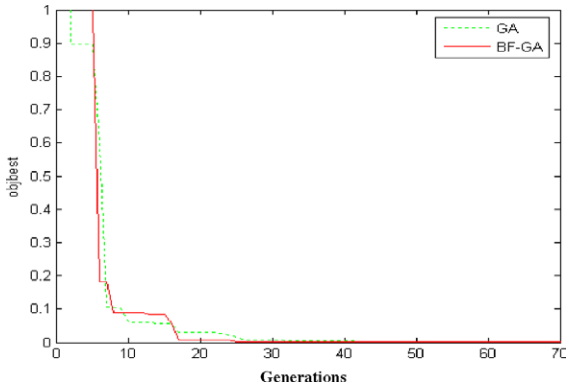

Fig. 5a. Convergence of GA and GA–BF for stepsize $= 1 \times 10^{-5}$ during the first 70 generations.

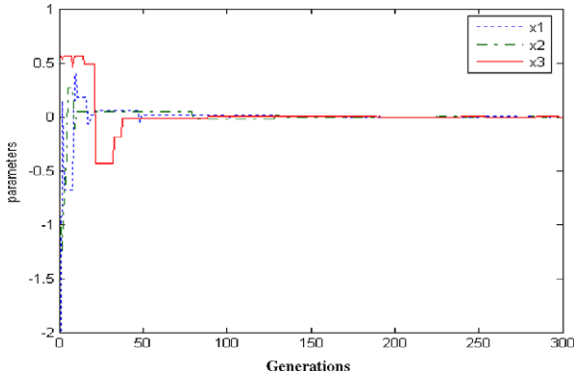

Fig. 5b. Tuning of parameters during 70 generations.

3.6. Performance of GA–BF for test functions

3.6.1. Test function: F_1

Figs. 5a and 5c illustrate the performance of GA and GA–BF with step size $= 1 \times 10^{-5}$ for 70 and 300 generations, respectively. As evident from Figs. 5a and 5c the hybrid GA–BF approach could search the optimal

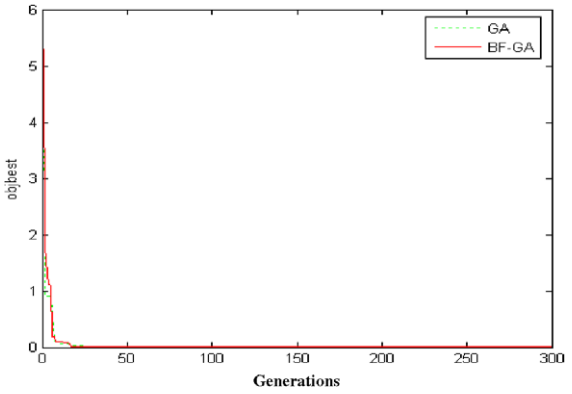

Fig. 5c. Convergence of GA and GA–BF for stepsize $= 1 \times 10^{-5}$ during 300 generations.

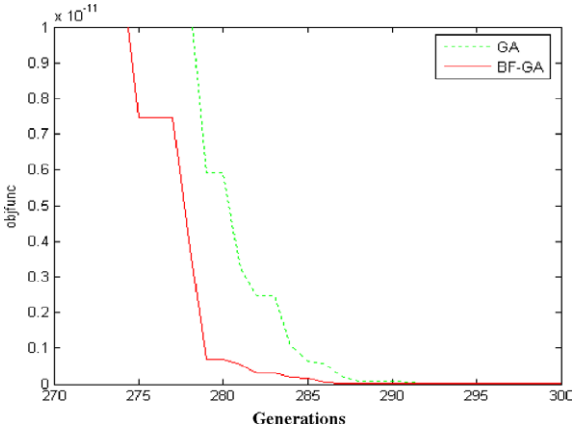

Fig. 5d. Performance of GA and GA–BF for stepsize $= 1 \times 10^{-5}$ during generations 270–300.

Table 4
Performance of GA and GA–BF

	x_1	x_2	x_3	Optimal objective function	Average objective function
GA	7.22E−08	5.07E−08	−9.43E−09	7.87E−15	8.03E−15
GA–BF	−1.70E−08	−1.44E−08	−2.31E−09	5.01E−16	1.43E−15

solutions earlier (10 generations) compared to a direct GA approach. Fig. 5b reveals that the GA–BF could converge faster than conventional GA during the final few iterations. Fig. 5d depicts how the parameters are optimized during the 27–300 generations by the GA and GA–BF for stepsize $= 1 \times 10^{-5}$. Table 4 depicts the final parameters values obtained using GA and GA–BF algorithms. Fig. 5e represents the characteristic of optimal variables during the 100 generations.

3.6.2. Test function: F_2

$$F_2(x) = 100\left(x_1^2 - x_2\right)^2 + (1 - x_1)^2.$$

Fig. 6a illustrates the contour of this function at $x = [1 \quad 1]^T$. Fig. 6b represents the performance characteristics of the conventional GA and the GA–BF algorithm.

From Fig. 6b, it is evident that the proposed GA–BF algorithm converges to the optimal solution much faster than the conventional GA approach. Table 5 illustrates the various empirical results obtained using GA and GA–BF approaches.

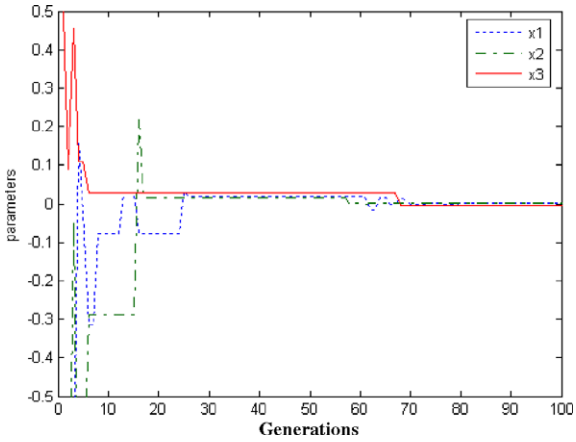

Fig. 5e. Tuning of parameters for stepsize $= 1 \times 10^{-5}$ during 100 generations.

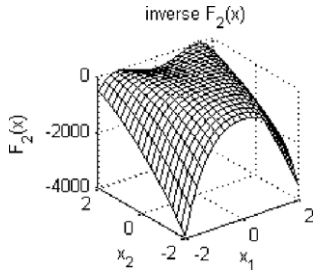

Fig. 6a. Contour of test function (F_2).

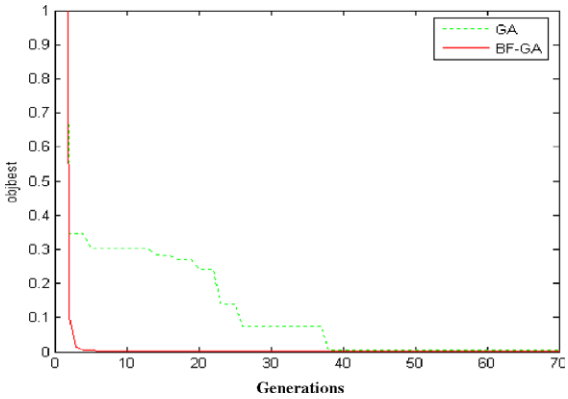

Fig. 6b. Performance of GA and GA–BF during the first 70 generations.

Table 5
GA and GA–BF performance for function F_2

	x_1	x_2	Optimal objective value	Average objective value
GA	0.001967	0.001967	1.0443267	1.0907699
BF–GA	5.12E−09	5.17E−09	0.9999285	0.9998567

3.6.3. Test function: F_3

$$F_3 = \sum_{i=1}^{5} [x_i].$$

This function has minimum $= -30$ at $x = [-5.12, -5.12, -5.12, -5.12, -5.12]$. Fig. 7a illustrates the contour map for this function and Figs. 7b–7d represent the various results obtained for F_3 and Table 6 illustrates the empirical performance.

3.6.4. Test function: F_4

Function $F_4 = \sum_{i=1}^{30} i x_i^4 + N(0,1)$ is used to compare the conventional GA and the proposed system GA–BF. Figs. 8a illustrates the contour map of this function. Figs. 8b, 8c depict the performance of GA and GA–BF method for different generation sizes. Figs. 8b and 8c illustrate that the proposed method converges faster than the conventional GA. Fig. 8d

3.6.5. Intelligent tuning of PID controller for automatic voltage regulator (AVR) using GA–BF approach

The transfer function of the PID controller for the AVR system is given by

$$\mathrm{PID}(s) = k_p + \frac{k_i}{s} + k_d s, \tag{8}$$

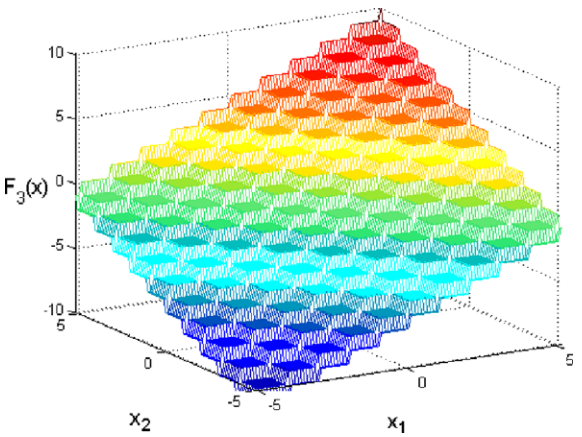

Fig. 7a. Contour map of test function F_3.

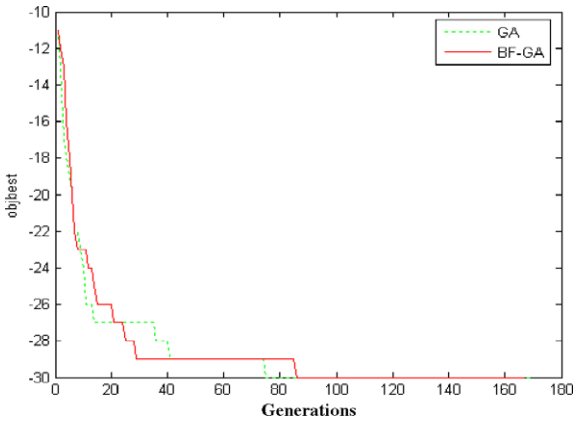

Fig. 7b. Performance of GA and GA–BF during the first 180 generations for test function F_3.

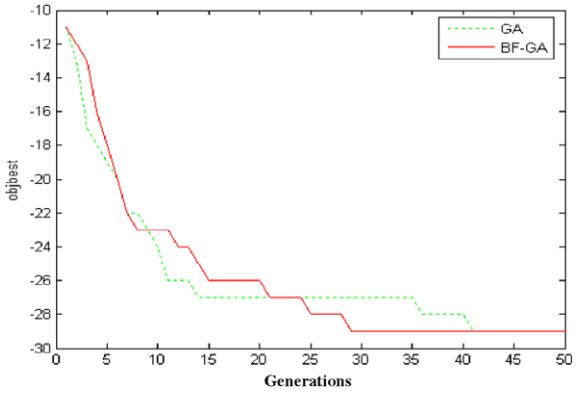

Fig. 7c. Performance of GA and GA–BF during the first 70 generations for test function F_3.

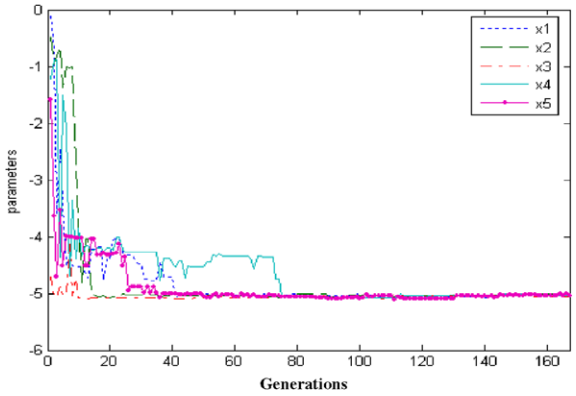

Fig. 7d. Tuning of parameters during 160 generations for test function F_3.

Table 6
Performance of GA and GA–BF for test function F_3

Method	x_1	x_2	x_3	x_4	x_5	Optimal objective value	Average objective value
GA	−5.024811	−5.015523	−5.059941	−5.03529	−5.03527	−30	−29.4
BF–GA	−5.111186	−5.097807	−5.089435	−5.06529	−5.06891	−30	−29.95

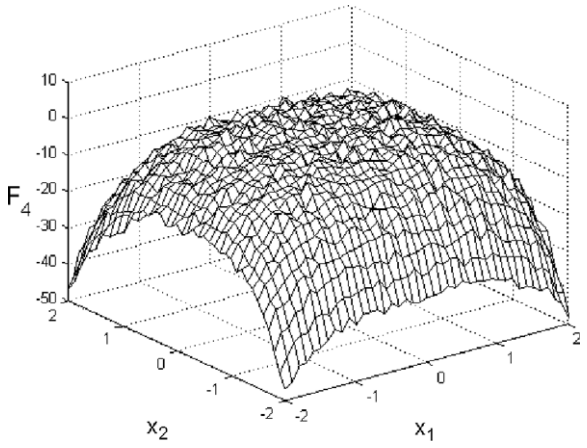

Fig. 8a. Contour map of test function F_4.

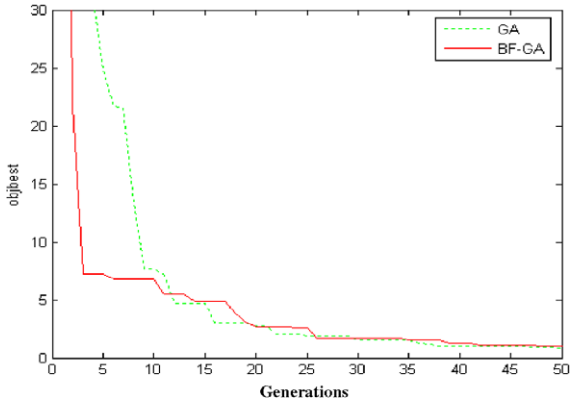

Fig. 8b. Performance of GA and GA–BF during the first 50 generations for test function F_4.

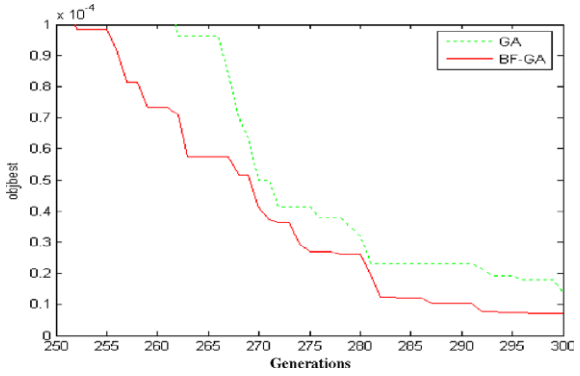

Fig. 8c. Performance of GA and GA–BF during generations 250–300 for test function F_4.

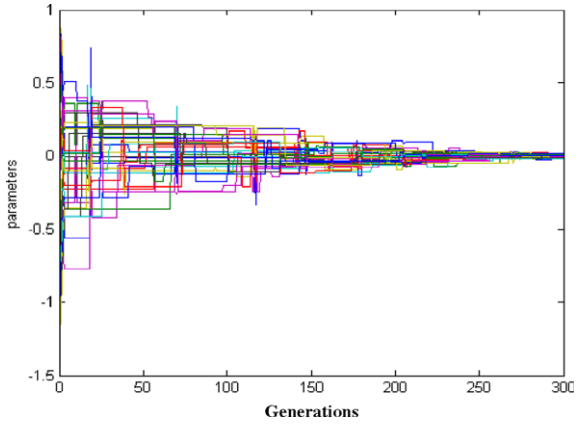

Fig. 8d. Tuning of parameters during 300 generations for test function F_4.

and the block diagram of the AVR system is shown in Fig. 9. The performance index of control response is defined by

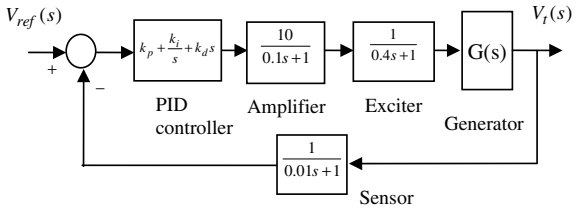

Fig. 9. Block diagram of an AVR system with a PID controller.

$$\begin{aligned}\min F(k_p,k_i,k_d) &= \frac{e^{-\beta\cdot t_s}/\max(t)}{(1-e^{-\beta})|1-t_r/\max(t)|} + e^{-\beta}\cdot Mo + ess \\ &= \frac{e^{-\beta}(t_s+\alpha_2\cdot||1-t_r/\max(t)|\cdot Mo|)}{(1-e^{-\beta})|1-t_r/\max(t)|} + ess = \frac{e^{-\beta}(t_s/\max(t)+\alpha\cdot Mo)}{\alpha} + ess \qquad (9)\\ \alpha &= \left(1-e^{-\beta}\right)\cdot|1-t_r/\max(t)|,\end{aligned}$$

k_p, k_i, k_d: Parameter of PID controller,
β: Weighting factor,
Mo: Overshoot,
t_s: Settling time (2%),
ess: Steady-state error,
t: Desired settling time.

In (9), if the weighing factor β increases, the rising time of response curve is small, and when β decreases, the rising time also increases. Performance criterion is defined as $Mo = 50.61\%$, $ess = 0.0909$, $t_r = 0.2693(s)$, $t_s = 6.9834(s)$. Initial values of the PID Controller and the GA–BF algorithm are depicted in Tables 7 and 8, respectively. For comparison purposes, we also used a Particle Swam Optimization (PSO) approach and a Hybrid GA–PSO approach [28].

The Particle Swarm Optimization (PSO) algorithm is mainly inspired by social behavior patterns of organisms that live and interact within large groups [11,38,6,27,24]. The standard PSO model consists of a swarm of particles, which are initialized with a population of random candidate solutions. They move iteratively through the d-dimension problem space to search the new solutions, where the fitness, f, can be calculated as a measure of certain qualities. Each particle has a position represented by a position-vector $\vec{x}_i$ (i is the index

Table 7
Range of PID parameters

PID parameters	Range	
	Min	Max
k_p	0	1.5
k_i	0	1
k_d	0	1

Table 8
Parameters of BF–GA algorithm

Parameters	Values
Stepsize	0.08
N_s	4
P_c	0.9
P_m	0.65

of the particle), and a velocity represented by a velocity-vector $\vec{v}_i$. Each particle remembers its own best position so far in a vector $\vec{x}_i^{\#}$, and the jth dimensional value of the vector $\vec{x}_i^{\#}$ is $x_{i,j}^{\#}$. The best position-vector among the swarm so far is then stored in a vector $\vec{x}^*$, and the jth dimensional value of the vector $\vec{x}^*$ is x_j^*. During the iteration time t, the update of the velocity from the previous velocity to the new velocity is determined and then the new position is determined by the sum of the previous position and the new velocity. The conventional PSO algorithm was used for controlling the mutation process of the genetic algorithm (GA), as an attempt to improve the GA learning efficiency. The architecture and flow chart of the proposed method are

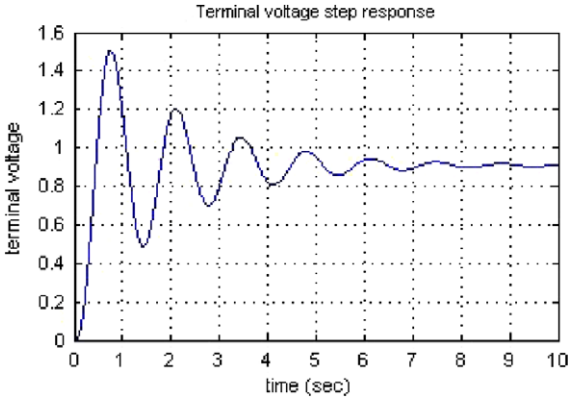

Fig. 10. Step response of terminal voltage in an AVR system without controller.

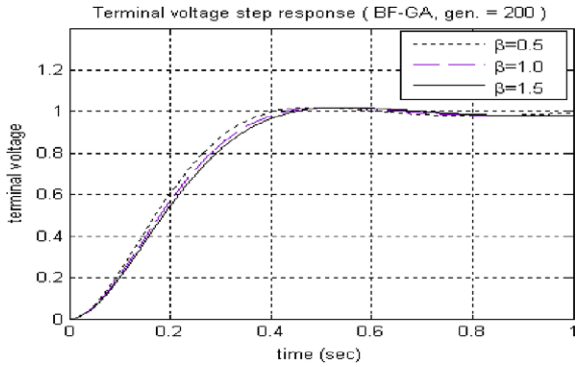

Fig. 11. Terminal voltage step response of an AVR system using GA–BF algorithm.

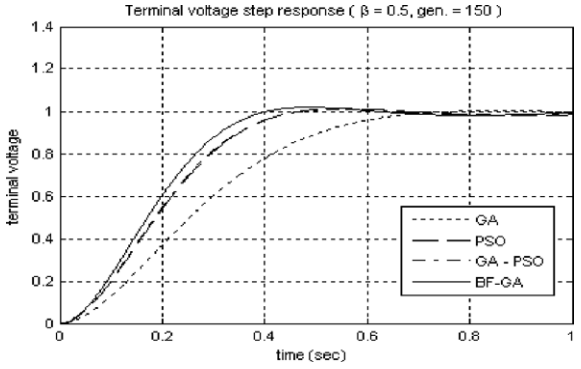

Fig. 12. Terminal voltage step response of an AVR system with different controllers ($\beta = 0.5$, generations = 200).

given in [26]. Euclidean distance is used for selecting crossover parents (in the hybrid GA–PSO approach) to avoid local optima and to obtain fast solutions.

Fig. 10 illustrates the response of terminal voltage for a step input in the control system. Figs. 11–14 represent the results obtained by GA and GA–BF algorithm for different β values for 200 generations as per Eq. (9). Figs. 15–17 illustrate the search process for optimal parameters for different β values (0.5, 1.0, and 1.5) by the GA–BF approach. Table 9 depicts the best solutions obtained using BF–GA controller for different β val-

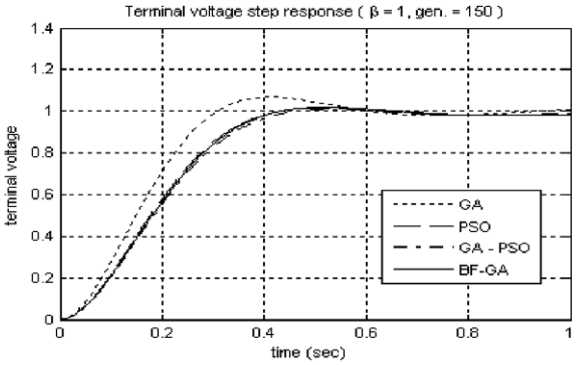

Fig. 13. Terminal voltage step response of an AVR system with different controllers ($\beta = 1.0$, generations = 200).

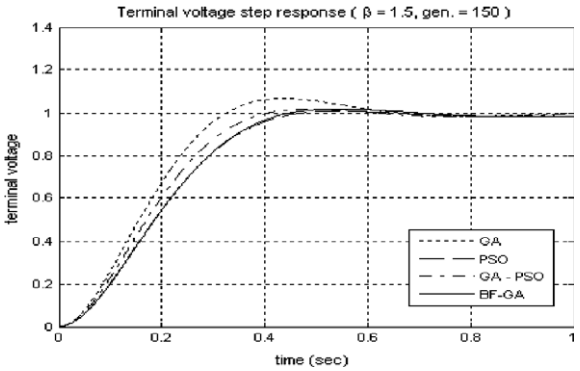

Fig. 14. Terminal voltage step response of an AVR system with different controllers ($\beta = 1.5$, generations = 200).

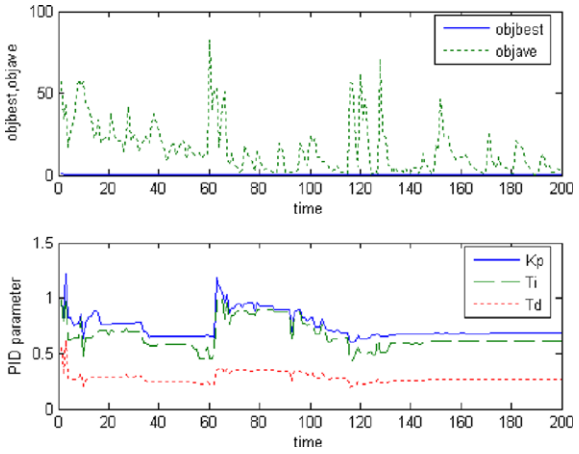

Fig. 15. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 0.5$.

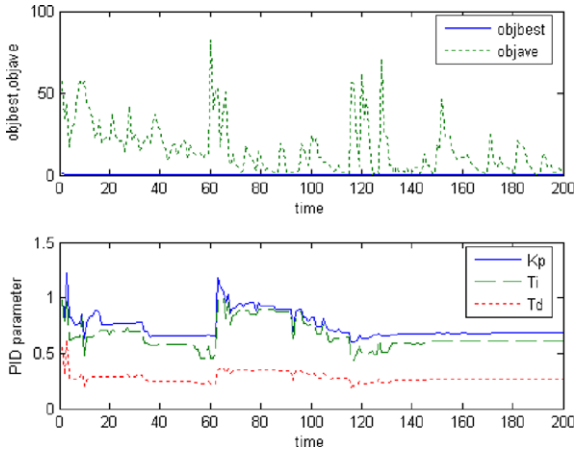

Fig. 16. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 1.0$.

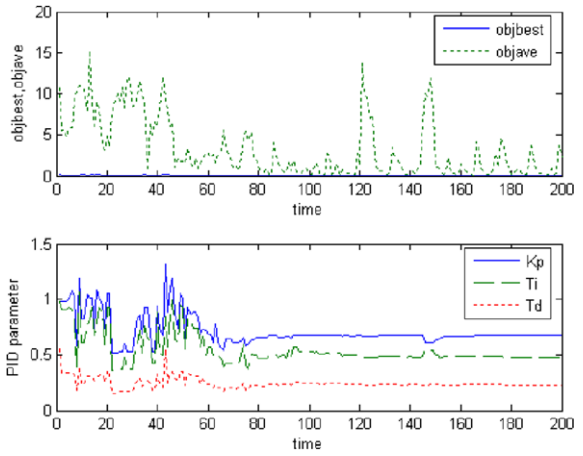

Fig. 17. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 1.5$.

Table 9
Best solutions obtained using BF–GA controller with different β values

β	Number of generation	k_p	k_i	k_d	Mo (%)	ess	t_s	t_r	Evaluation value
0.5	200	0.68233	0.6138	0.26782	1.94	0.0171	0.3770	0.2522	0.3614
1	200	0.68002	0.52212	0.24401	1.97	0.0067	0.4010	0.2684	0.1487
1.5	200	0.67278	0.47869	0.22987	1.97	0.0014	0.4180	0.2795	0.07562

Table 10
Comparison of the objective value using different methods ($\beta = 1.5$, generation = 200)

β	Methods	k_p	k_i	k_d	Mo (%)	ess	t_s	t_r	Evaluation value
1.5	GA	0.8282	0.7143	0.3010	6.7122	0.0112	0.5950	0.2156	0.0135
	PSO	0.6445	0.5043	0.2348	0.8399	0.0084	0.4300	0.2827	0.0073
	GA–PSO	0.6794	0.6167	0.2681	1.8540	0.0178	0.8000	0.2526	0.0071
	BF–GA	0.6728	0.4787	0.2299	1.97	0.0014	0.4180	0.2795	0.0756

ues and Table 10 illustrates a performance comparison of the values obtained using different methods ($\beta = 1.5$, 200 generations). For all the experiments, we have used a fixed number of generations which was decided by

trial and error. To make the algorithm more adaptive, the best scheme would be to choose a stopping criterion once the improvement in the solution does not justify the number of generations required for it [15].

4. Conclusions

Recently many variants of genetic algorithms have been investigated for improving the learning and speed of convergence. For some problems, the designer often has to be satisfied with local optimal or suboptimal solutions.

This paper proposed a novel hybrid approach consisting of a GA (genetic algorithm) and BF (Bacterial Foraging) and the performance is illustrated using various test functions. Also, the proposed GA–BF algorithm is used for tuning a PID controller of AVR system. As evident from the graphical and empirical results, the suggested hybrid system GA–BF performed very well. Our future research would include the analysis and performance of the PID controller in the presence of output noise or input disturbances.

The proposed approach has potential to be useful for other practical optimization problems (e.g., engineering design, online distributed optimization in distributed computing and cooperative control) as social foraging models work very well in such environments.

Acknowledgement

Authors thank the Editor-in-Chief and the three anonymous reviewers for the constructive comments which helped to improve the clarity and presentation of the paper.

References

[1] A. Abraham, EvoNF: A framework for optimization of fuzzy inference systems using neural network learning and evolutionary computation, in: The 17th IEEE International Symposium on Intelligent Control, ISIC'02, IEEE Press, ISBN 0780376218, 2002, pp. 327–332.
[2] J. Alcock, Animal Behavior, An Evolutionary Approach, Sinauer Associates, Sunderland, Massachusetts, 1998.
[3] P. Angelov, A fuzzy controller with evolving structure, Information Sciences 161 (1–2) (2004) 21–35.
[4] J. Arabas, Z. Michalewicz, J. Mulawka, GAVaPS – A genetic algorithm with varying population size, in: Proceedings IEEE International Conference on Evolutionary Computation, Orlando, 1994, pp. 73–78.
[5] W.J. Bell, Searching Behavior, The Behavioral Ecology of Finding Resources, Chapman and Hall, London, England, 1991.
[6] E. Bonaneau, M. Dorigo, G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems, Oxford University Press, New York, NY, 1999.
[7] A.L. Buczak, R.E. Uhrig, Hybrid fuzzy – genetic technique for multisensor fusion, Information Sciences 93 (3–4) (1996) 265–281.
[8] D.R. Carvalho, A.A. Freitas, A hybrid decision tree/genetic algorithm method for data mining, Information Sciences 163 (1–3) (2004) 13–35.
[9] P. Chootinan, A. Chen, Constraint handling in genetic algorithms using a gradient-based repair method, Computers and Operations Research 33 (8) (2006) 2263–2281.
[10] M. Dotoli, G. Maione, D. Naso, E.B. Turchiano, Genetic identification of dynamical systems with static nonlinearities, in: Proceedings IEEE SMC Mountain Workshop Soft Computing Industrial Applications, Blacksburg, VA, 2001, pp. 65–70.
[11] R. Eberchart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of International Symposium on Sym. Micro Machine and Human Science, Nagoya, Japan, 1995, pp. 39–43.
[12] J.D. Farmer, N.H. Packard, A.S. Perelson, The immune system, adaptation, and machine learning, Physica D 22 (1986) 187–204.
[13] P.J. Fleming, R.C. Purshouse, Evolutionary algorithms in control system engineering: A survey, Control Engineering Practice 10 (2002) 1223–1241.
[14] C.M. Fonseca, P.J. Fleming, Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: A unified formulation; – Part II: Application example, IEEE Transactions on Systems Man and Cybernetics Part A – Systems and Humans 28 (1) (1998) 26–47.
[15] S.K. Foo, P. Saratchandran, N. Sundararajan, An evolutionary algorithm for parallel mapping of backpropagation learning on heterogeneous processor networks, International Journal of Systems Science 30 (3) (1999) 309–321.
[16] V. Gazi, K.M. Passino, Stability analysis of social foraging swarms, IEEE Transactions on Systems Man and Cybernetics Part B – Cybernetics 34 (1) (2004) 539–557.
[17] Z.L. Gaing, A particle swarm optimization approach for optimum design of pid controller in AVR system, IEEE Transactions on Energy Conversion 19 (2) (2004) 384–391.
[18] T. Glad, L. Ljung, Control Theory, Multivariable and Nonlinear Methods, Taylor and Francis, London, UK, 2000.

[19] A.F. Gómez-Skarmeta, M. Valdés, F. Jiménez, J.G. Marín-Blázquez, Approximative fuzzy rules approaches for classification with hybrid-GA techniques, Information Sciences 136 (1–4) (2001) 193–214.
[20] G.J. Gray, D.J. Murray-Smith, Y. Li, K.C. Sharman, T. Weinbrenner, Nonlinear model structure identification using genetic programming, Control Engineering Practice 6 (1998) 1341–1352.
[21] D. Grunbaum, Schooling as a strategy for taxis in a noisy environment, Evolutionary Ecology 12 (1998) 503–522.
[22] M. Haouari, J.C. Siala, A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem, Computers and Operations Research 33 (5) (2006) 1274–1288.
[23] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions on Systems Man and Cybernetics Part B 34 (2004) 997–1006.
[24] J. Kennedy, R. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, Inc., San Francisco, CA., 2001.
[25] K. Kristinnson, G.A. Dumont, System identification and control using genetic algorithms, IEEE Transactions on Systems Man and Cybernetics 22 (1992) 1033–1046.
[26] D.H. Kim, J.H. Cho, Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization, in: Piotr S. Szczepaniak, Janusz Kacprzyk, Adam Niewiadomski (Eds.), Third International Atlantic Web Intelligence Conference, AWIC 2005, Lodz, Poland, Lecture Notes in Computer Science, Vol. 3528, Advances in Web Intelligence, 2005, pp. 231–238.
[27] D.H. Kim, J.H. Cho, Intelligent control of AVR system using GA–BF, in: Rajiv Khosla, Robert J. Howlett, Lakhmi C. Jain (Eds.), Proceeding of KES 2005, Melbourne, Australia, Lecture Notes in Computer Science, Vol. 3684/2005, 2005, pp. 854–860.
[28] D.H. Kim, J.I. Park, Intelligent tuning of PID controller for AVR system using a hybrid GA–PSO approach, Lecture Notes in Computer Science 3645/2005 (2005) 366–373.
[29] R.A. Krohling, J.P. Rey, Design of optimal disturbance rejection PID controllers using genetic algorithms, IEEE Transactions on Evolutionary Computation 5 (2001) 78–82.
[30] Z.J. Lee, C.Y. Lee, A hybrid search algorithm with heuristics for resource allocation problem, Information Sciences 173 (1–3) (2005) 155–167.
[31] C.L. Lin, H.W. Su, Intelligent control theory in guidance and control system design: An overview, Proceedings of the National Science Council, Republic of China – Part A: Physical Science and Engineering 24 (1) (2000) 15–30.
[32] B. Maione, D. Naso, B. Turchiano, GARA: A genetic algorithm with resolution adaptation for solving system identification problems, in: Proceedings European Control Conference, Porto, Portugal, 2001, pp. 3570–3575.
[33] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, New York, 1999.
[34] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin Heideberg, 1996.
[35] K. Mori, M. Tsukiyama, Immune algorithm with searching diversity and its application to resource allocation problem, Transactions on JIEE 113-C (10) (1993).
[36] K.M. Passino, Biomimicry of Bacterial foraging for Distributed Optimization, University Press, Princeton, New Jersey, 2001.
[37] K.M. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Systems Magazine (2002) 52–67.
[38] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.
[39] D.W. Stephens, J.R. Krebs, Foraging Theory, Princeton University Press, Princeton, New Jersey, 1986.
[40] R. Tanese, Distributed genetic algorithm, in: Proceedings of International Conference on Genetic Algorithms, 1989, pp. 434–439.
[41] S. Tsutsui, D.E. Goldberg, Simplex crossover and linkage identification: Single-stage evolution vs. multi-stage evolution, in: Proceedings IEEE International Conference on Evolutionary Computation, 2002, pp. 974–979.
[42] H. Yoshida, K. Kawata, Y. Fukuyama, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems 15 (2000) 1232–1239.