8

A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization and Robust Tuning of PID Controller with Disturbance Rejection

D.H. Kim and A. Abraham

Summary. The social foraging behavior of *Escherichia coli* (*E. Coli*) bacteria has been used to solve optimization problems. This chapter proposes a hybrid approach involving genetic algorithm (GA) and bacterial foraging (BF) algorithm 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). To design disturbance rejection tuning, disturbance rejection conditions based on H_∞ are illustrated and the performance of response is computed for the designed PID controller as the integral of time weighted squared error. Simulation results clearly illustrate that the proposed approach is very efficient and could easily be extended for other global optimization problems.

8.1 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.

In the past, some researchers have focused on using hybrid genetic algorithm approaches for optimization problems. Buczak and Uhrig [1] 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.

Gomez-Skarmeta et al. [3] 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.

D.H. Kim and A. Abraham: *A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization and Robust Tuning of PID Controller with Disturbance Rejection*, Studies in Computational Intelligence (SCI) **75**, 171–199 (2007)
www.springerlink.com

To discover classification rules, Carvalho and Freitas [2] 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 [4] 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 [5] 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 [6] 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 [7, 8]. After many generations, poor foraging strategies are either eliminated or restructured 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 [9].

The rest of the chapter is organized as follows. Section 8.2 provides a brief literature overview of the bacterial foraging algorithm followed by the proposed hybrid approach based on bacterial foraging (BF) and genetic algorithms (GA). The performance of the algorithm is illustrated using four benchmark functions in Sect. 8.3 [10]. The proposed hybrid algorithm is further validated for PID controller tuning in Sect. 8.4. PID controller tuning with disturbance rejection is presented in Sect. 8.5. Some conclusions are also provided towards the end.

8.2 Hybrid System Consisting of Genetic Algorithm and Bacteria Foraging

8.2.1 Genetic Algorithms

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.

8.2.2 Bacterial Foraging Algorithm

Recently, search and optimal foraging of bacteria have been used for solving optimization problems [7]. 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.

Overview of Chemotactic Behavior of *Escherichia coli*

We considered the foraging behavior of *E. coli*, which is a common type of bacteria. Its behavior and movement comes from a set of six rigid spinning (100–200 rps) flagella, each driven as a biological motor. An *E. coli* bacterium alternates through running and tumbling. Running speed is 10–20 μm s^{-1} , 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 [8].

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 semisolid 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.

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

$$H(i,j,k) = \phi^x(j,k,l)| \quad x = 1,2,\ldots,N. \tag{8.1}$$

(8.1) 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 lth bacterium at position $\phi^x(i,j,k)$ in R^n, and

$$\phi^x = (i+1,j,k) = \phi^x(i,j,k) + C(x)\varphi(i), \tag{8.2}$$

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 $P_c^i(\phi) \quad i = 1,2,\ldots,S$ to model the cell-to-cell signaling via an attractant and a repellant is represented by Passino [8]:

$$\begin{aligned} P_c(\phi) &= \sum_{i=1}^{N} P_{cc}^i \\ &= \sum_{i=1}^{N} \left[-L_{\text{attract}} \exp\left(-\delta_{\text{attract}} \sum_{j=1}^{n} (\phi_j -)^2 \right) \right] \\ &+ \sum_{i=1}^{N} \left[-K_{\text{repellant}} \exp\left(-\delta_{\text{attract}} \sum_{j=1}^{n} (\phi_j - \phi_j^i)^2 \right) \right], \end{aligned} \tag{8.3}$$

where $\phi = [\phi_1, \cdots, \phi_p]^T$ is a point on the search space, L_{attract} is the depth of the attractant released by the cell and σ_{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 σ_{attract} is a measure of the width of the repellant. The expression P_ϕ 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 $P_{\rm ar}(\phi)$ to represent the environment-dependent cell-to-cell signaling as

$$P_{\rm ar}(\phi) = \exp(T - P(\phi))P_c(\phi), \tag{8.4}$$

where T is a tunable parameter. By considering the minimization of $P(i,j,k,l) + P_{\rm ar}(\phi^i(j,kl))$, 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 $P_{\rm ar}(\phi^i(j,k,l))$ implies that, with T being constant, the smaller the value of $P(\phi)$, the lager $P_{\rm ar}(\phi)$ 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, $P_{\rm ar}(\phi)$ is much larger than J_ϕ, 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 $P_{\rm ar}(\phi)$ is much smaller than $P(\phi)$, and it is the effect of the nutrients that dominates. In $P_{\rm ar}(\phi)$, the scaling factor of $P_{\rm c}(\phi)$ is given as in exponential form.

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

[**Step 1**] Initialize parameters $n, N, N_{\rm c}, N_{\rm s}, N_{\rm re}, N_{\rm ed}, P_{\rm ed}, C(i)(\quad i = 1,2,\cdots,N), \phi^i$. where,

n: Dimension of the search space,
N: The number of bacteria in the population,
$N_{\rm c}$: Chemotactic steps,
$N_{\rm re}$: The number of reproduction steps,
$N_{\rm ed}$: The number of elimination-dispersal events,
$P_{\rm ed}$: 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: $j = j + 1$

[**Sub-step A**] For $i = 1,2,\cdots,N$, take a chemotactic step for bacterium i as follows.

[**Sub-step B**] Compute fitness function, integral of time weighted squared error, ITSE(i,j,k,l).

[**Sub-step C**] Let $\text{ITSE}_{\rm last} = \text{ITSE}(i,j,k,l)$ to save this value, since we may find a better cost during a run.

[**Sub-step D**] Tumble: generate a random vector $\Delta(i)\varepsilon R^n$ with each element $\Delta_m(i), \quad m = 1,2,\cdots,p$, a random number within $[-1,1]$.

[**Sub-step 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.

[**Sub-step F**] Compute ITSE$(i,j+1,k,l)$.

[**Sub-step G**] Swim

(1) Let $m = 0$ (counter for swim length).
(2) 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 in [Sub-step F].
Else, let $m = N_s$. This is the end of the while statement.

[**Sub-step H**] Go to next bacterium (i, j) if $i \neq N$(i.e., go to [Sub-step 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:

[**Sub-step A**] For the given k and l, and for each $i = 1, 2, \cdots, N$, 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).

[**Sub-step B**] The S, bacteria with the highest $\mathrm{ITSE}_{\mathrm{health}}$ values die and the remaining S, bacteria with the best values split (this process is performed by the copies that are made placed at the same location as their parent).

[**Step 7**] If $k < N_{\mathrm{re}}$, 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 $i = 1, 2, \cdots, N$, 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 $l < N_{\mathrm{ed}}$, then go to [Step 2]; otherwise end.

8.3 Experiment Results Using Test Functions

This section illustrates some comparisons between the proposed GA–BF (genetic algorithm–bacteria foraging algorithm) and the conventional Simple Genetic Algorithm (SGA) using some test functions as depicted in Table 8.1. Table 8.1 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).

8.3.1 Mutation Operation in GA–BF

Dynamic mutation [11] is used in the proposed GA–BF algorithm using

$$x_j = \begin{cases} \tilde{x}_j + \Delta(k, x_j^{(\mathrm{U})} - \tilde{x}_j) & \tau = 0 \\ \tilde{x}_j - \Delta(k, \tilde{x}_j - x_j^{(\mathrm{L})}) & \tau = 1, \end{cases} \tag{8.5}$$

Table 8.1. Initial conditions for test functions and variation of different parameters

Test function	Range		GA parameters				BF 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

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

$$\Delta(k,y) = y\eta(1 - \frac{k}{z})^A. \tag{8.6}$$

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

8.3.2 Crossover Operation in GA–BF

A modified simple crossover [12] is used for the BF-GA algorithm using

$$\tilde{x}_j^u = \lambda \bar{x}_j^v + (1-\lambda)\bar{x}_j^u \tag{8.7}$$

$$\tilde{x}_j^v = \lambda \bar{x}_j^u + (1-\lambda)\bar{x}_j^v, \tag{8.8}$$

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.

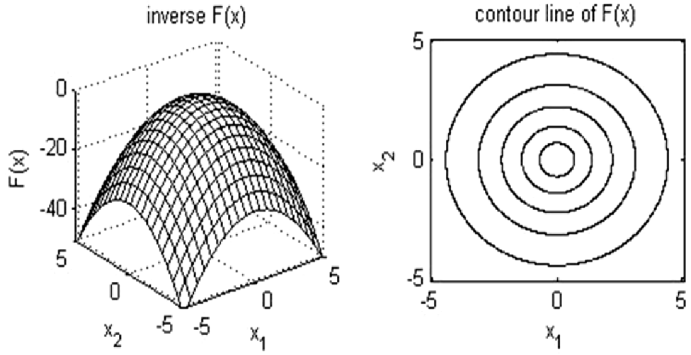

Fig. 8.1. Contour of test function F_1

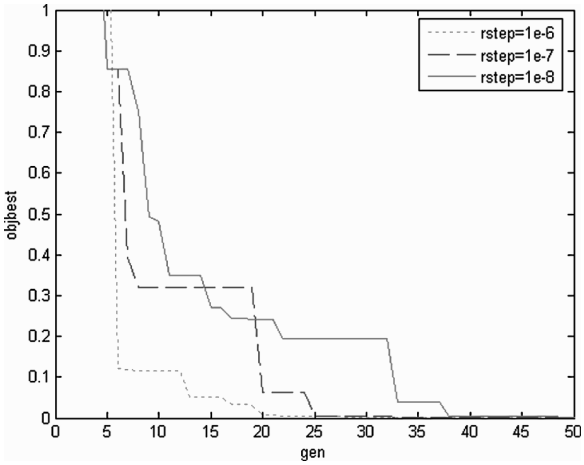

Fig. 8.2. Performance value for the three different step sizes for the first 50 generations

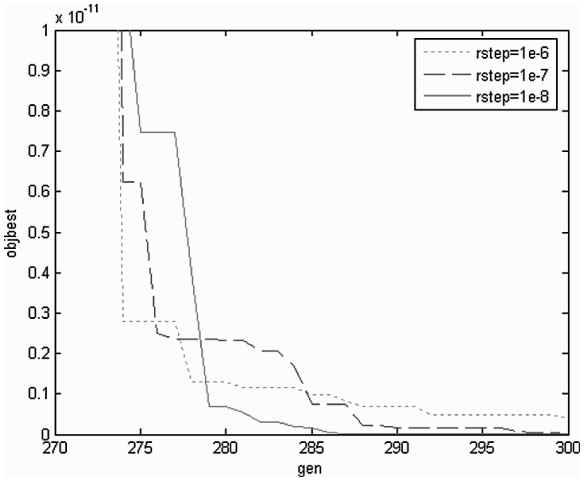

Fig. 8.3. Performance value for the three different step sizes for generations 270–300

8.3.3 Performance Variation for Different Step Sizes

Step size refers to the moving distance per step of the bacteria. For performance comparison, test function (F_1)is used as depicted in Fig. 8.1. Figures 8.2 and 8.3 illustrate the performance of the GA–BF algorithm for 300 generations. As evident from the results, for bigger step sizes, the convergence is faster. Table 8.2 illustrates the empirical performance.

$$F(x) = \sum_{i=1}^{3} x_i^2, \quad -5.12 \leq x_1, x_2, x_3 \leq 5.11. \tag{8.9}$$

Table 8.2. Parameter values for various step sizes

Step size	x_1	x_2	x_3	Optimal obj. function	Average obj. 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

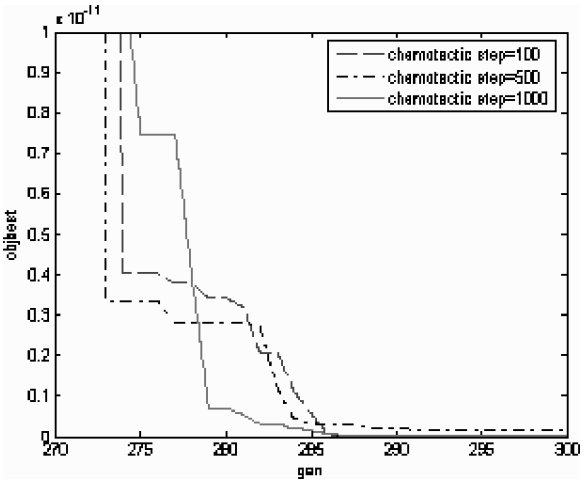

Fig. 8.4. Performance value for different chemotactic step sizes for generations 270–300

Table 8.3. Variation of objective function values for different chemotactic steps

CS size	x_1	x_2	x_3	Optimal obj. function	Average obj. function
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
1,000	−1.70E−08	−1.44E−08	−2.31E−09	5.01E−16	1.43E−15

8.3.4 Performance for Different Chemotactic Steps of GA–BF

Figure 8.4 and Table 8.3 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.

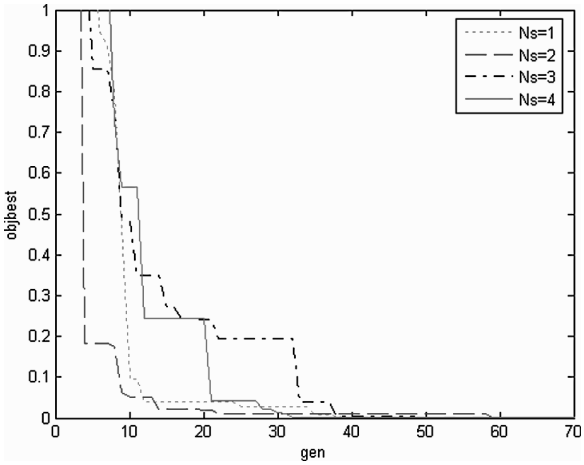

Fig. 8.5. Performance value for different lifetime (N_s) for the first 70 generations

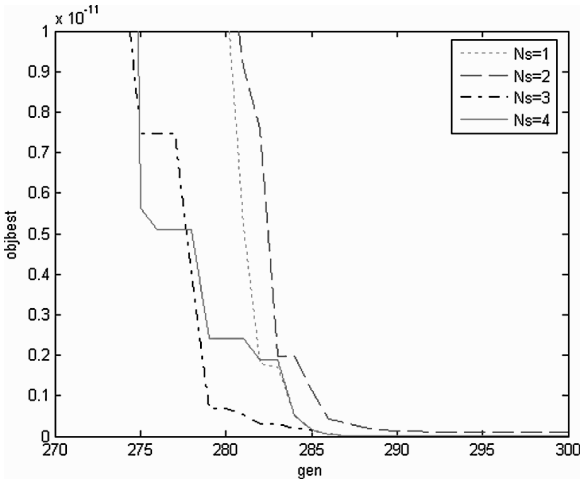

Fig. 8.6. Performance value for different lifetime (N_s) for generations 270–300

8.3.5 Performance for Different Life Time (N_s)

Figures 8.5 and 8.6 illustrate the characteristics between objective function and the number of generations for different life time (N_s) of the bacteria.

8.3.6 Performance of GA–BF for Test Functions

Test Function $F_1(x) = \sum_{i=1}^{3} x_i^2$

Figures 8.7, 8.9 and 8.10 illustrate the performance of GA and GA–BF for step size = 1×10^{-5} for 1–300 generations. As evident, the hybrid GA–BF approach could

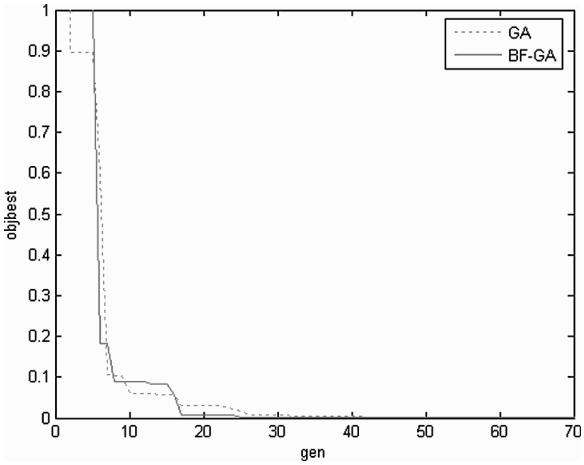

Fig. 8.7. Convergence of GA and GA–BF for stepsize=1×10^{-5} during the first 70 generations

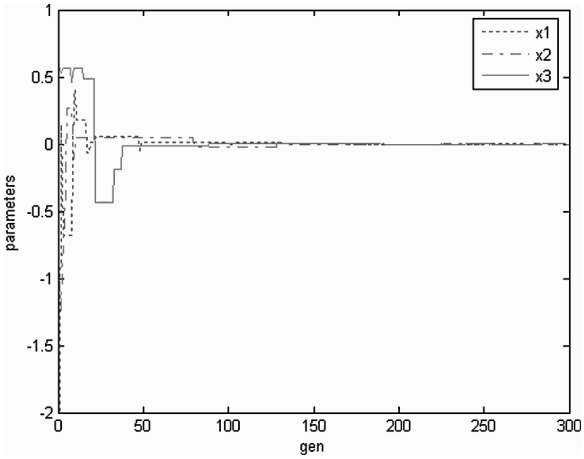

Fig. 8.8. Tuning of parameters during 70 generations

search the optimal solutions earlier (10 generations) compared to a direct GA approach. It also reveals that the GA–BF could converge faster than conventional GA during the final few iterations. Table 8.4 depicts the final parameters values obtained using GA and GA–BF algorithms. Figures 8.8 and 8.11 represent the characteristics of optimal variables during the 100 generations.

Test Function $F_2(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2$

Figure 8.12 illustrates the contour of this function at $x = [1,1]^T$. Figure 8.13 represents the performance characteristics of the conventional GA and the GA–BF algo-

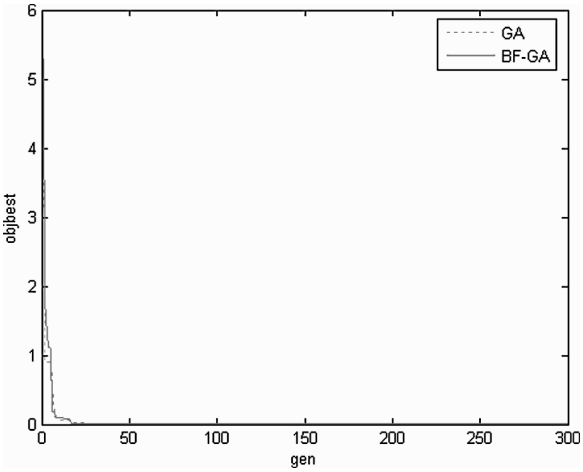

Fig. 8.9. Convergence of GA and GA–BF for stepsize=1×10^{-5} during 300 generations

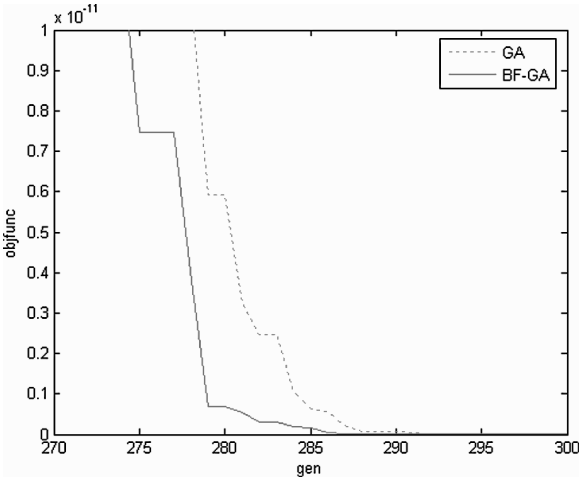

Fig. 8.10. Performance of GA and GA–BF for stepsize=1×10^{-5} during generations 270–300

Table 8.4. Performance of GA and GA–BF

	x_1	x_2	x_3	Optimal obj. function	Average obj. 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

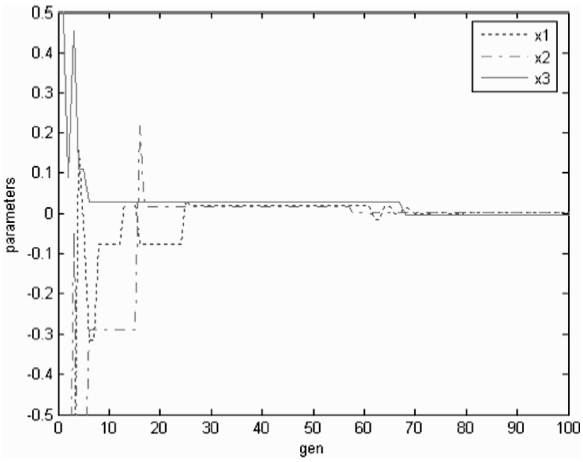

Fig. 8.11. Tuning of parameters for stepsize=1×10^{-5} during 100 generations

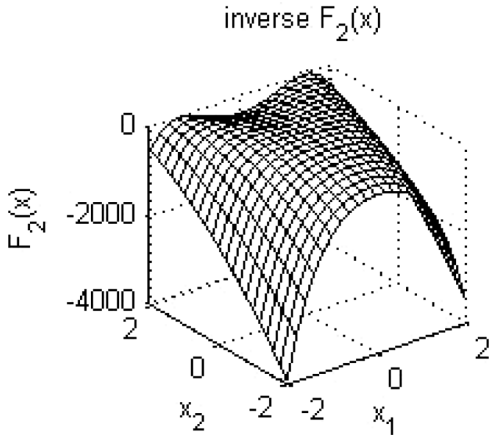

Fig. 8.12. Contour of test function F_2

rithm. From Fig. 8.13, it is evident that the proposed GA–BF algorithm converges to the optimal solution much faster than the conventional GA approach. Table 8.5 illustrates the various empirical results obtained using GA and GA–BF approaches.

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

This function has *minimum* $= -30$ at $x = [-5.12, -5.12, -5.12, -5.12, -5.12,]$. Figure 8.14 illustrates the contour map for this function and Figs. 8.15–8.17 represent the various results obtained for F_3 and Table 8.6 illustrates the empirical performance.

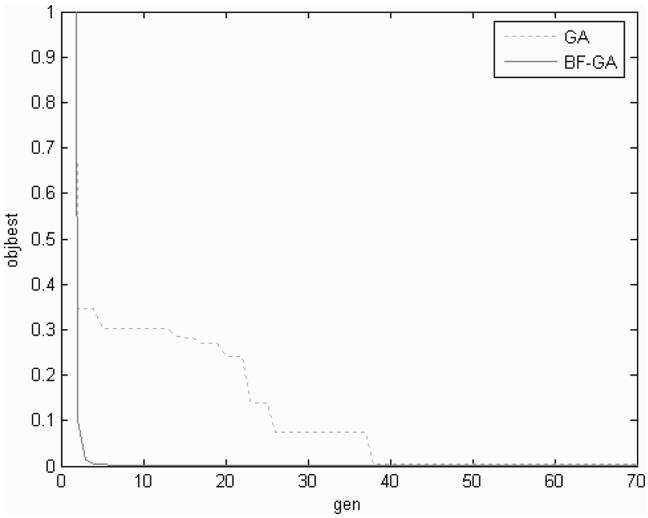

Fig. 8.13. Performance of GA and GA–BF during the first 70 generations

Table 8.5. 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

Test Function $F_4 = \sum_{i=1}^{30} ix_i^4 + N(0,1)$

Figure 8.18 illustrates the contour map of this function. Figures 8.19–8.22 depict the performance of GA and GA–BF method for different generation sizes.

8.4 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

$$\mathrm{PID}(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s} + k_\mathrm{d} s. \tag{8.10}$$

and the block diagram of the AVR system is shown in Fig. 8.23. Step response of terminal voltage in an AVR system without controller is depicted in Fig. 8.24. The performance index of control response is defined by

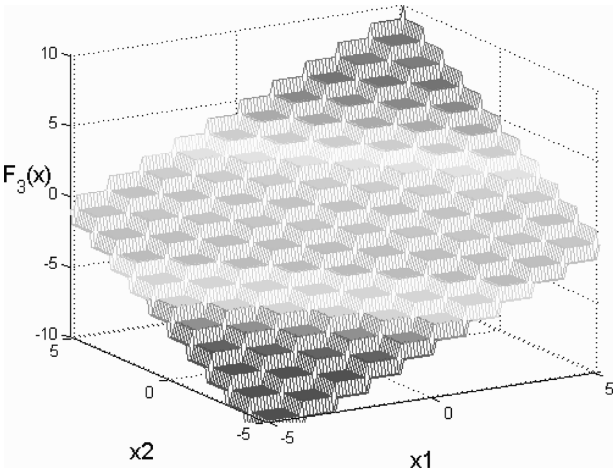

Fig. 8.14. Contour map of test function F_3

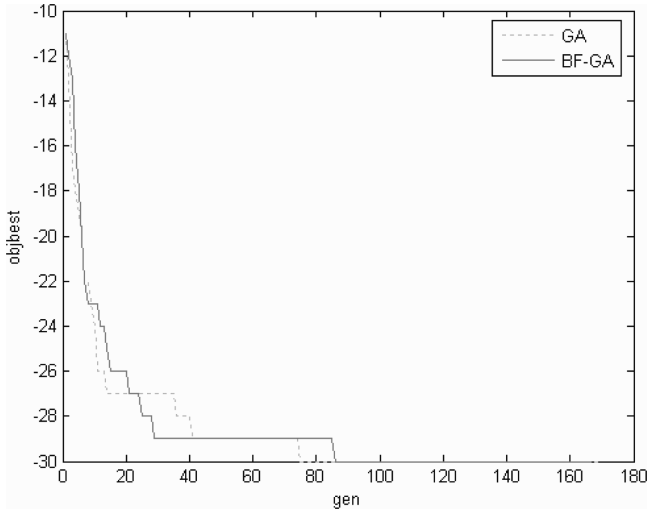

Fig. 8.15. Performance of GA and GA–BF during the first 180 generations for F_3

$$\begin{aligned}\min F(k_{\mathrm{p}}, k_{\mathrm{i}}, k_{\mathrm{d}}) &= \frac{\mathrm{e}^{-\beta} t_{\mathrm{s}} / \max(t)}{\left(1-\mathrm{e}^{-\beta}\right)|1-t_{\mathrm{r}}/\max(t)|} + \mathrm{e}^{-\beta}\mathrm{Mo} + \mathrm{ess} \\ &= \frac{\mathrm{e}^{-\beta}(t_{\mathrm{s}} + \alpha_2|1-t_{\mathrm{r}}/\max(t)\mathrm{Mo}|)}{(1-\mathrm{e}^{-\beta}|1-t_{\mathrm{r}}/\max(t)|)} + \mathrm{ess} \\ &= \frac{\mathrm{e}^{-\beta}\left(t_{\mathrm{s}}/\max(t) + \alpha\mathrm{Mo}\right)}{\alpha} + \mathrm{ess},\end{aligned} \tag{8.11}$$

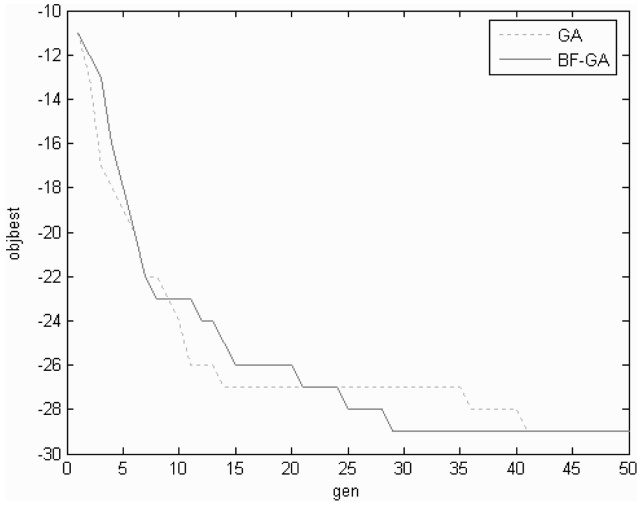

Fig. 8.16. Performance of GA and GA–BF during the first 50 generations for F_3

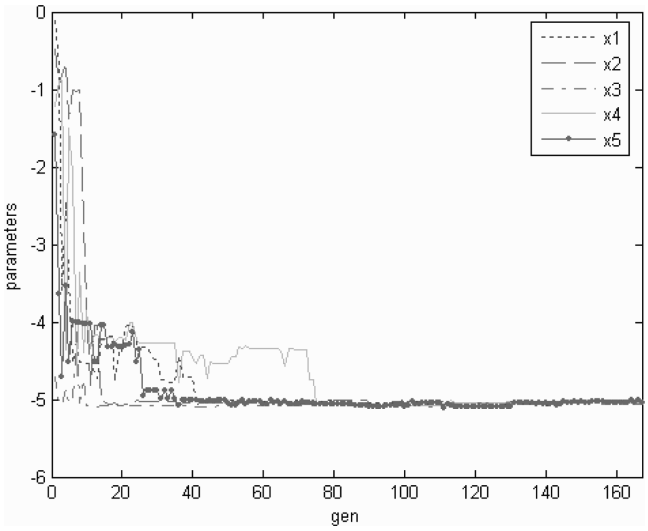

Fig. 8.17. Tuning of parameters during 160 generations for F_3

where $\alpha = (1 - e^{-\beta})\,|1 - t_r/\max(t)|$, k_p, k_i, k_d the parameters of PID controller; β the weighting factor; Mo the overshoot; t_s the settling time (2%); ess the steady-state error; t is the desired settling time.

In (8.11), if the weighing factor 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, $t_r = 0.2693(s)$, and $t_s = 6.9834(s)$. Initial

Table 8.6. Performance of GA and GA–BF for test function F_3

Method	x_1	x_2	x_3	x_4	x_5	Opt. obj. val.	Avg. obj. val.
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

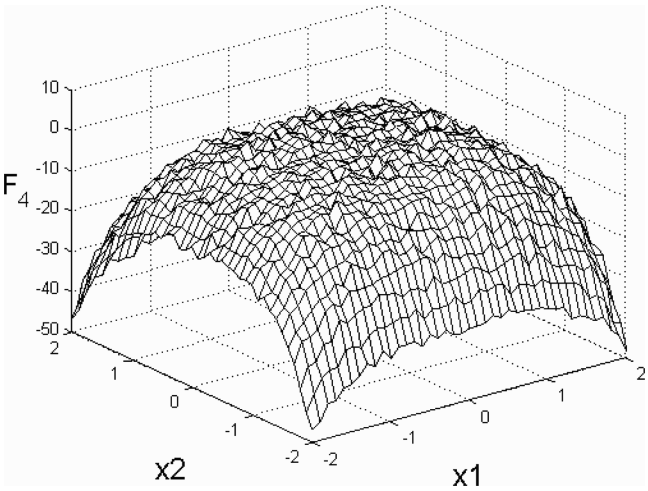

Fig. 8.18. Contour map of test function F_4

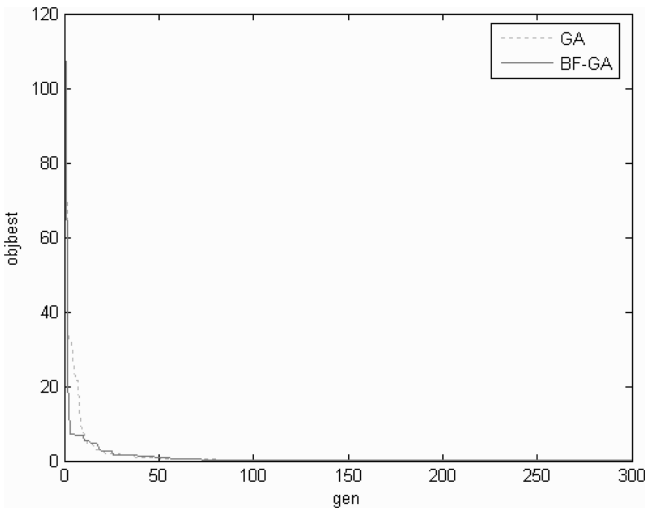

Fig. 8.19. Performance of GA and GA–BF during 300 generations for F_4

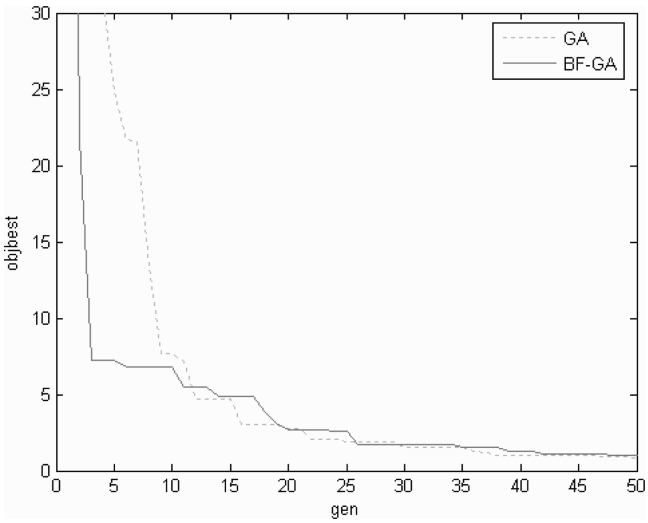

Fig. 8.20. Performance of GA and GA–BF during the first 50 generations for F_4

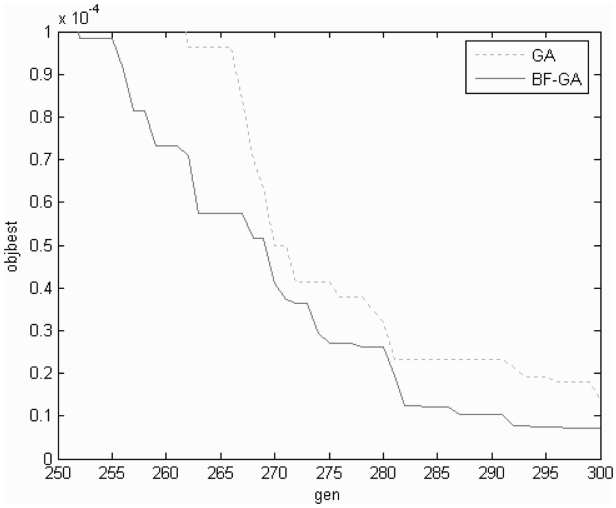

Fig. 8.21. Performance of GA and GA–BF during generations 250–300 for F_4

values of the PID Controller and the GA–BF algorithm are depicted in Tables 8.7 and 8.8, respectively. For comparison purposes, we also used a particle swarm optimization (PSO) approach [14] and a hybrid GA-PSO approach [10]. As given in the previous chapter, Euclidean distance is used for selecting crossover parents (in the hybrid GA-PSO approach) to avoid local optima and to obtain fast solutions.

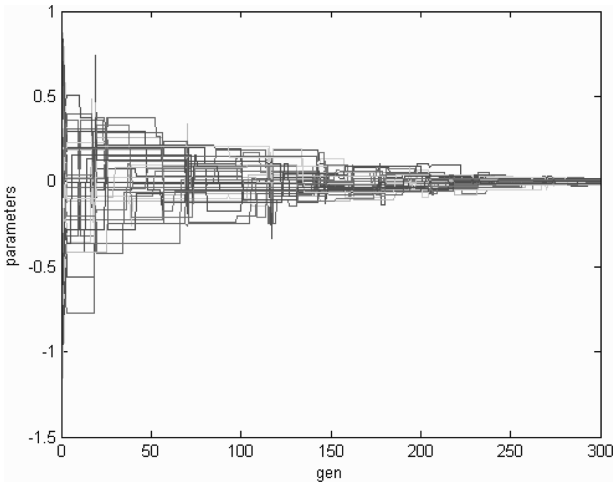

Fig. 8.22. Tuning of parameters during 300 generations for F_4

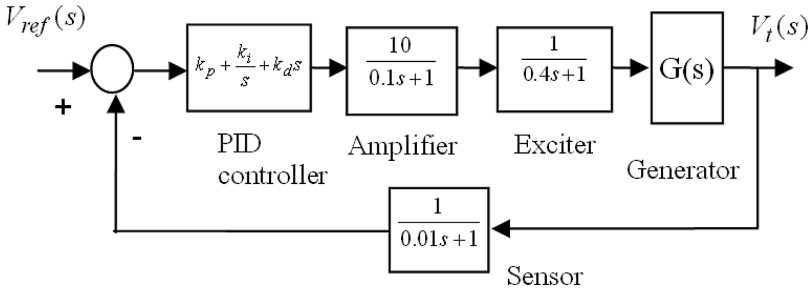

Fig. 8.23. Block diagram of an AVR system with a PID controller

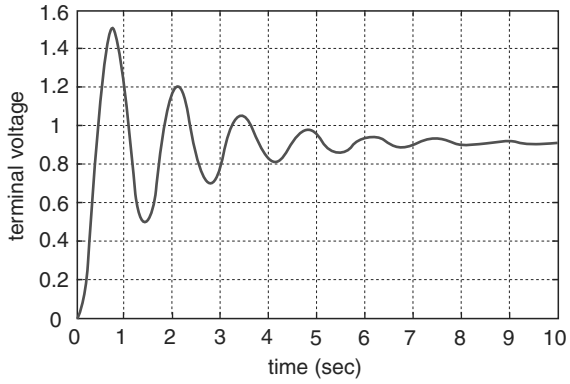

Fig. 8.24. Step response of terminal voltage in an AVR system without controller

Table 8.7. Range of PID parameters

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

Table 8.8. Parameters of the BF-GA algorithm

Parameters	Values
Stepsize	0.08
N_s	4
P_c	0.9
P_m	0.65

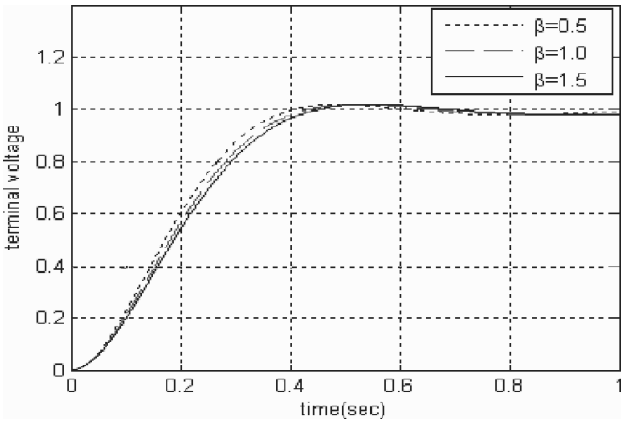

Fig. 8.25. Terminal voltage step response of an AVR system using GA–BF algorithm

Figures 8.25–8.31 represent the results obtained by GA and GA–BF algorithm for different β values for 200 generations as per (8.11). Table 8.9 depicts the best solutions obtained using BF-GA controller for different β values and Table 8.10 illustrates a performance comparison of the values obtained using different methods (β =1.5, 200 generations). For all the experiments, we have used a fixed number of generations, which was decided by trial and error (Figs. 8.32 and 8.33).

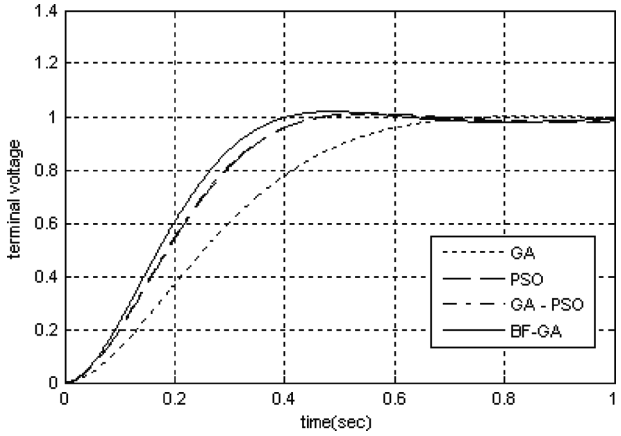

Fig. 8.26. Terminal voltage step response of an AVR system with different controllers ($\beta =$ 0.5, generations = 200)

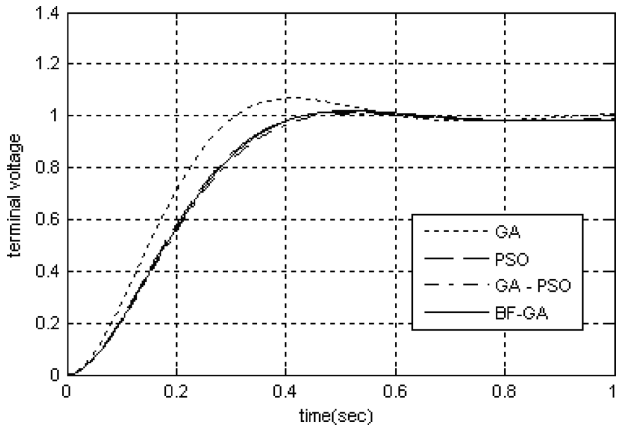

Fig. 8.27. Terminal voltage step response of an AVR system with different controllers ($\beta =$ 1.0, generations = 200

8.5 PID Controller Tuning With Disturbance Rejection Function

8.5.1 Condition for Disturbance Rejection

With reference to Fig. 8.32, the disturbance rejection constraint is given by Xu et al. [15, 16].

$$\max_{d(t)\varepsilon D} \frac{\|Y\|}{\|d\|} = \left\| \frac{w(s)}{1+K(s,c)G(s)} \right\|_{\infty} < \delta. \tag{8.12}$$

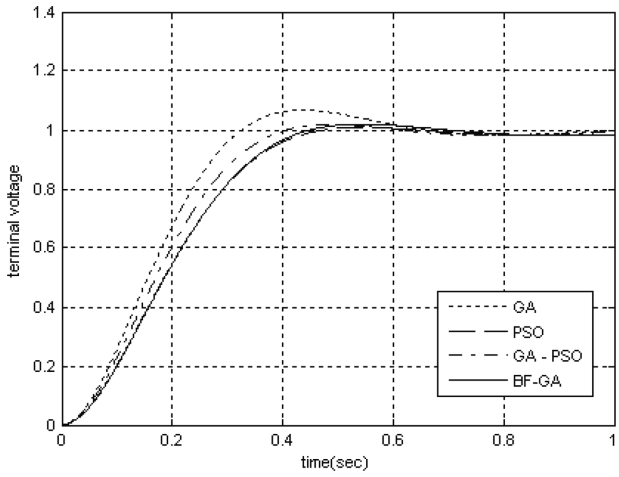

Fig. 8.28. Terminal voltage step response of an AVR system with different controllers ($\beta =$ 1.5, generations $= 200$

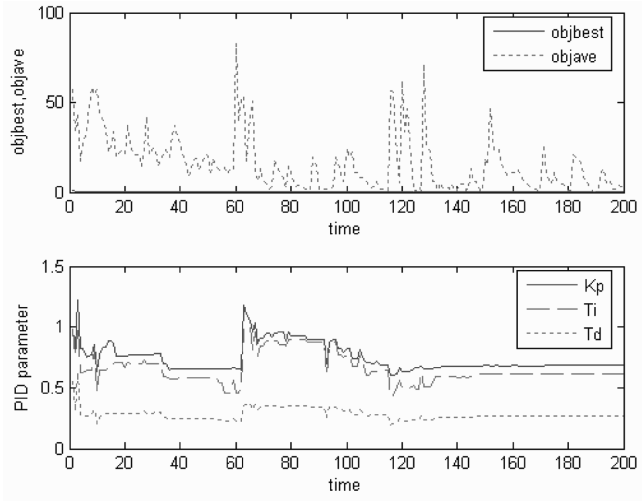

Fig. 8.29. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 0.5$

where $\delta < 1$ is a constant defined by the desired rejection level and $\|\cdot\|_\infty$ denotes the H_∞-norm, which is defined as

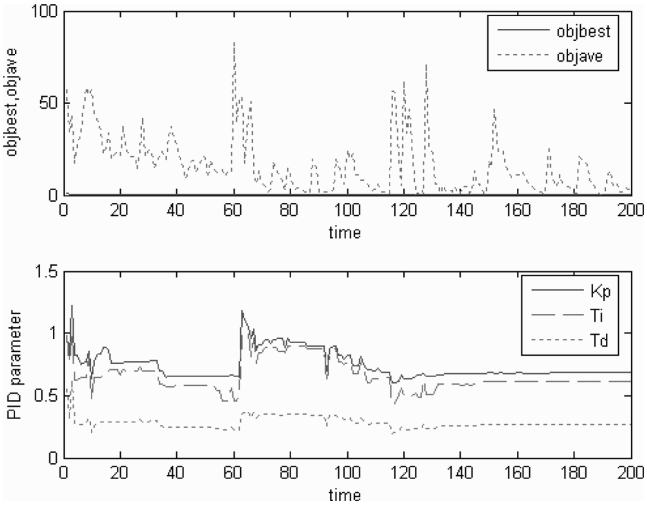

Fig. 8.30. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 1.0$

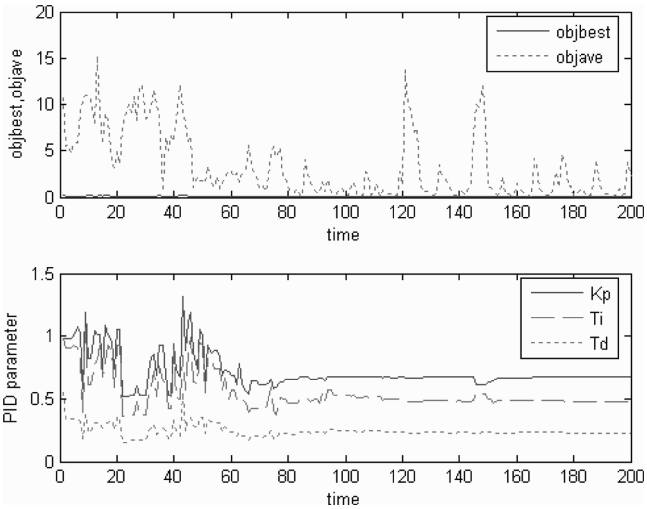

Fig. 8.31. Search process for optimal parameter values of an AVR system by GA–BF method for $\beta = 1.5$

Then the disturbance rejection constraint becomes

$$\left\| \frac{\omega(s)}{1+K(s,c)G(s)} \right\|_{\infty} = \max_{\omega\varepsilon[0,)} \left(\frac{\omega(\mathrm{j}\omega)\omega(-\mathrm{j}\omega)}{1+K(\mathrm{j}\omega,c)G(\mathrm{j}\omega,c)K(-\mathrm{j}\omega,c)G(-\mathrm{j}\omega,c)} \right)^{0.5} = \max_{\omega\varepsilon[0,)} (\sigma(\omega,c))^{0.5}. \tag{8.14}$$

Table 8.9. Performance obtained using BF-GA controller for different β values

β	Generations	k_p	k_i	k_d	Mo (%)	ess	t_s	t_r	Eval. 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 8.10. Comparison of the objective value using different methods ($\beta = 1.5$, generation=200)

β	Method	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

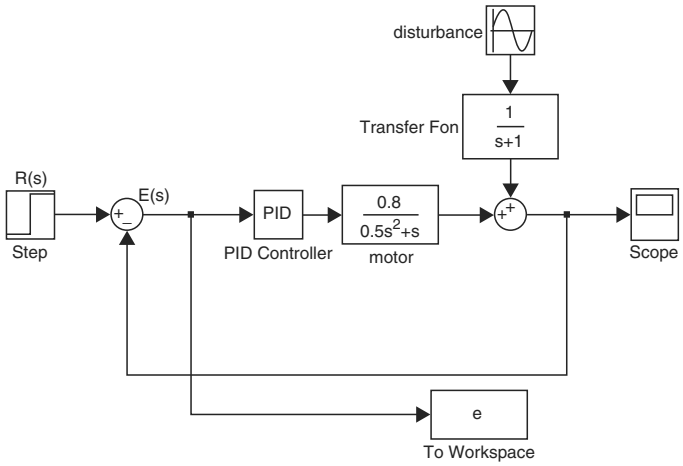

Fig. 8.32. Control system with disturbance

The controller $K(s,c)$ is denoted as

$$K(s,c) = c_1 + \frac{c_2}{s} + c_3 s. \tag{8.15}$$

The vector c of the controller parameter is given by

$$c = [c_1, c_2, c_3]^T. \tag{8.16}$$

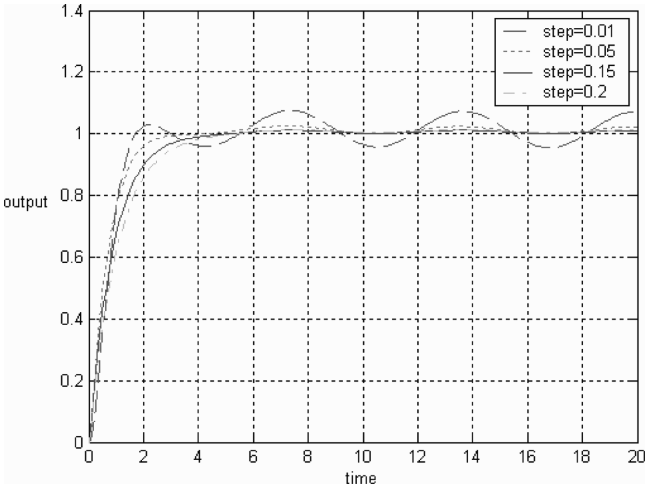

Fig. 8.33. Step response by variation of chemotactic step size

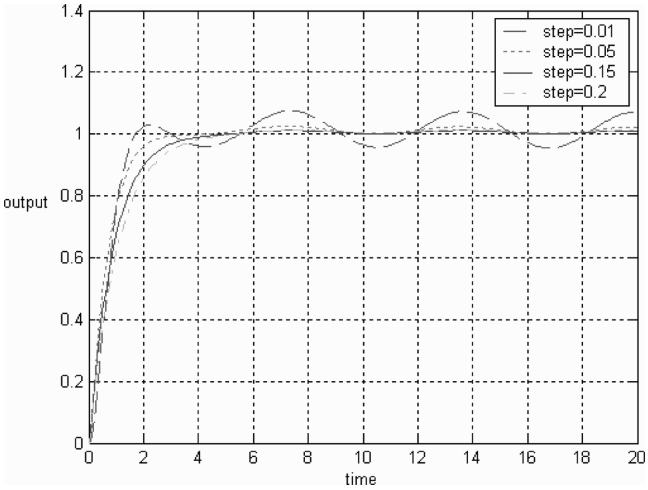

Fig. 8.34. Comparison of different methods

Hence, the condition for disturbance rejection is given as

$$\max_{\omega \varepsilon [0,)} (\sigma(\omega, c))^{0.5} < \delta.$$

8.5.2 Performance Index for Disturbance Rejection Controller Design

The performance index is defined as integral of the time-weighted square of the error (ITSE) and is given by

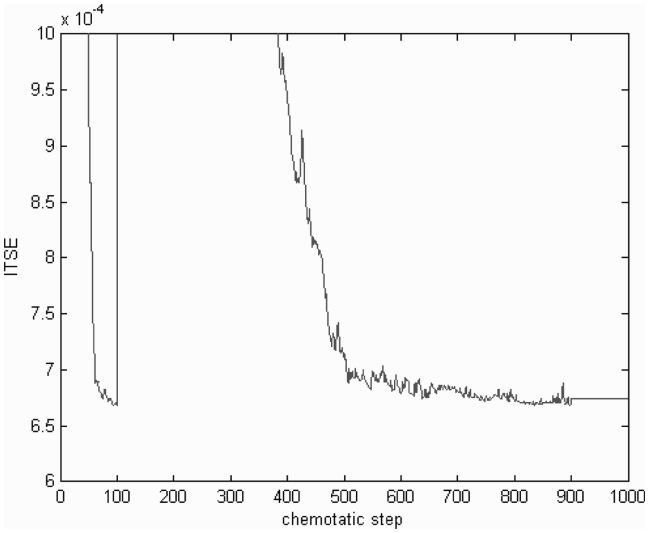

Fig. 8.35. Improvement of performance index (ITSE) by GA–BF approach

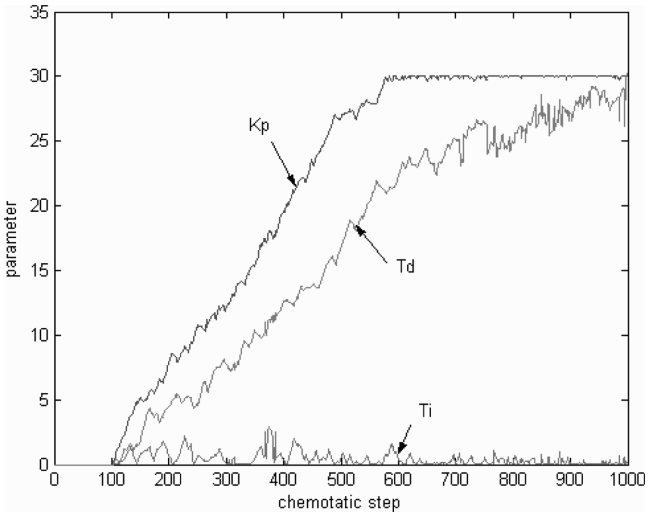

Fig. 8.36. Search process of optimal PID parameters by GA–BF

$$\mathrm{PI} = \int_0^\infty t(E(t))^2 \,\mathrm{d}t, \tag{8.17}$$

$$E(s) = \frac{B(s)}{A(s)} = \frac{\sum_{j=0}^{m} b_j s^{m-1}}{\sum_{i=0}^{n} a_i s^n}. \tag{8.18}$$

$E(s)$ contains the parameters of the controller (c) and plant, the value of performance index (PI) for a system of nth order can be minimized by adjusting the vector

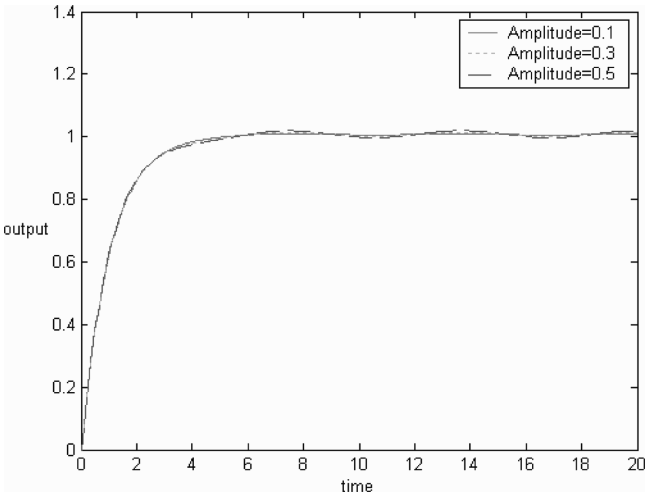

Fig. 8.37. Step response to a type of sine wave disturbance

Table 8.11. PID parameters and ITSE for different chemotactic step sizes

CS size	ITSE	K_p	T_i	T_d
$CS_{size} = 0.01$	0.094163	3.0605	0.076235	1.1411
$CS_{size} = 0.05$	0.003656	13.704	0.2733	8.773
$CS_{size} = 0.15$	0.000678	30.000	0.23208	25.844
$CS_{size} = 0.2$	0.000668	29.901	0.25813	30.000

c as follows [16]:

$$\min_{c} \mathrm{PI}(c). \tag{8.19}$$

For optimal tuning, the task is to find the vector c, such that the ITSE performance index (PI(c)) is a minimum using the hybrid GA–BF algorithm and the constraint $\max_{\omega\varepsilon[0,\infty)}(\sigma)(\omega,c)^{0.5} < \delta$ is satisfied.

8.5.3 Simulations and Discussions

Figure 8.33 illustrates the step response to variation of chemotactic size. The best response was obtained for step size = 0.15. Figure 8.34 depicts a comparison of results using GA, artificial immune system (AIS) [17], and hybrid GA–BF approach. Figure 8.35 is representing search process of performance index (ITSE) by GA–BF and Fig. 8.36 depicts the search process to have optimal PID parameters. Figure 8.37 illustrates the step response to a type of sine wave disturbance (Tables 8.11 and 8.12).

Table 8.12. Comparison of PID parameters and ITSE using different methods

	GA–BF	GA[1]	AIS
K_p	29.901	29.992	29.739
T_i	0.25813	0.0001	0.39477
T_d	301	28.3819	27.277
ITSE	0.000668	0.000668	0.0006352

8.6 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 chapter 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 with disturbance rejection function. Simulation results illustrate satisfactory responses. As evident from the graphical and empirical results, the suggested hybrid system GA–BF performed very well.

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.

8.7 Acknowledgements

Authors would like to thank the anonymous reviewers for the constructive comments which helped to improve the clarity and presentation of the Chapter.

References

1. Buczak AL and Uhrig RE (1996), Hybrid fuzzy-genetic technique for multisensor fusion, Information Sciences, Volume 93, Issues 3–4, pp. 265–281.
2. Carvalho DR and Freitas AA (2004), A hybrid decision tree/genetic algorithm method for data mining, Information Sciences, Volume 163, Issues 1–3, pp. 13–35.
3. Gomez-Skarmeta AF, Valdes M, Jimenez F, and Marín-Blázquez JG (2001), Approximative fuzzy rules approaches for classification with hybrid-GA techniques, Information Sciences, Volume 136, Issues 1–4, pp. 193–214.
4. Lee ZJ and Lee CY (2005), A hybrid search algorithm with heuristics for resource allocation problem, Information Sciences, Volume 173, Issues 1–3, pp. 155–167.
5. Chootinan P and Chen A (2006), Constraint handling in genetic algorithms using a gradient-based repair method, Computers and Operations Research, Volume 33, Issue 8, pp. 2263–2281.

6. Haouari M and Siala JC (2006), A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem, Computers and Operations Research, Volume 33, Issue 5, pp. 1274–1288.
7. Passino KM (2001), Biomimicry of Bacterial Foraging for Distributed Optimization, University Press, Princeton, New Jersey.
8. Passino KM (2002), Biomimicry of Bacterial Foraging for Distributed Optimization and Control, IEEE Control Systems Magazine, pp. 52–67.
9. Kim DH and Cho JH (2005), Intelligent Control of AVR System Using GA–BF, Proceeding of KES 2005, Melbourne, Australia, Lecture Notes in Computer Science, Volume 3684/2005, Editors: Khosla R, Howlett RJ, Jain LC, pp. 854–860.
10. Kim DH, Abraham A, and Cho JH (2007), Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization, Information Sciences, Elsevier Science, 2007. $http://dx.doi.org/10.1016/j.ins.2007.04.002$
11. Michalewicz Z (1999), Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York.
12. Michalewicz Z (1996), Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin Heideberg.
13. Kim DH and Park JI (2005) , Intelligent Tuning of PID Controller For AVR System Using a Hybrid GA-PSO Approach, Lecture Notes in Computer Science, Volume 3645/2005, pp. 366–373.
14. Eberchart R and Kennedy J (1995), A New Optimizer Using Particle Swarm Theory, Proceedings of the International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39–43.
15. Xu JX, Pok YM, Liu C, and Hang CC (1998), Tuning and analysis of a fuzzy PI controller based on gain and phase margins, IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans, Volume 28, Issue 5, pp. 685–691.
16. Xu JX, Liu C, and Hang CC (1996), Tuning of Fuzzy PI Controllers Based on Gain/Phase Margin Specifications and ITAE Index, ISA Transactions, Volume 35, pp. 79–91.
17. Farmer JD, Packard NH, and Perelson AS (1986), The immune system, adaptation, and machine learning, Physica D: Nonlinear Phenomena, Volume 22, pp. 187–204.