7

Hybrid Genetic: Particle Swarm Optimization Algorithm

D.H. Kim, A. Abraham, and K. Hirota

Summary. This chapter proposes a hybrid approach by combining a Euclidian distance (EU) based genetic algorithm (GA) and particle swarm optimization (PSO) method. The performance of the hybrid algorithm is illustrated using four test functions. Proportional integral derivative (PID) controllers have been widely used in industrial systems such as chemical process, biomedical process, and in the main steam temperature control system of the thermal power plant. Very often, it is difficult to achieve an optimal PID gain without prior expert knowledge, since the gain of the PID controller has to be manually tuned by a trial and error approach. Using the hybrid EU–GA–PSO approach, global and local solutions could be simultaneously found for optimal tuning of the controller parameters.

7.1 Introduction

During the last decade, genetic algorithm-based approaches have received increased attention from the engineers dealing with problems, which could not be solved using conventional problem solving techniques. A typical task of a GA in this context is to find the best values of a predefined set of free parameters associated with either a process model or a control vector. A possible solution to a specific problem can be encoded as an individual (or a chromosome), which consists of 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 an evaluating mechanism to obtain its fitness value. Using this fitness value and genetic operators, a new population is generated iteratively which is referred to as a generation. The GA uses the basic reproduction operators such as crossover and mutation to produce the genetic composition of a population. Many efforts for the enhancement of conventional genetic algorithms have been proposed. Among them, one category focuses on modifying the structure of the population or on the individual's role while another category is focused on modification/efficient control of the basic operations, such as crossover or mutation, of conventional genetic algorithms [9].

The proportional integral derivative (PID) controller has been widely used owing to its simplicity and robustness in chemical process, power plant, and electrical systems [1]. Its popularity is also due to its easy implementation in hardware and software. However, using only the P, I, D parameters, it is often very difficult to control a plant with complex dynamics, such as large dead time, inverse response, and for power plants having a high nonlinear characteristics [5]. Recently, there has been a growing interest in the usage of intelligent approaches such as fuzzy inference systems, neural network, evolutionary algorithms, and their hybrid approaches for the tuning of a PID controller [1–4, 6, 7].

This chapter introduces a hybrid approach consisting of genetic algorithm and particle swarm optimization (PSO) algorithm. To obtain an advanced learning structure, there are two processing steps in the proposed method. In the first step, Euclidean distance is used to select the global data for crossover and mutation operators to avoid local minima, and to obtain fast convergence. In the second step, in order to enhance the learning efficiency of GA, PSO strategy is applied. The proposed approach focuses on the advantage of PSO into the mutation process of GA, for improving the GA learning efficiency. A PSO like search proceeds through the problem space, with the moving velocity of each particle represented by a velocity vector. Therefore, global and local optimal solution can be simultaneously achieved and the most appropriate parameter of the PID controller can be selected for the given plant and system [11].

We first illustrate the performance of the proposed hybrid approach using four test functions. Further the performance of hybrid EU–GA–PSO approach is validated by tuning a PID controller of a automatic voltage regulator (AVR). The chapter is organized as follows: In Sect. 7.2, we introduce the hybrid approach using Euclidean distance-based genetic algorithm and PSO algorithm with some simple illustrations. Detailed experiment results for function optimization are illustrated in Sect. 7.3 followed by PID controller tuning in Sect. 7.4. Some Conclusions are also provided in the end.

7.2 Hybrid Approach Using Euclidean Distance Genetic Algorithm and Particle Swarm Optimization Algorithm

7.2.1 Particle Swarm Optimization Algorithm

The PSO algorithm conducts search using a population of particles which correspond to individuals in a genetic algorithm [8, 10]. A population of particles is initially randomly generated. Each particle represents a potential solution and has a position represented by a position vector. A swarm of particles moves through the problem space, with the moving velocity of each particle represented by a velocity vector. At each time step, a function representing a quality measure is calculated by using as input. Each particle keeps track of its own best position, which is associated with the best fitness it has achieved so far in a vector. Furthermore, the best position among all the particles obtained so far in the population is kept track as output. In addition to this global version, another local version of PSO keeps track of the best position among all the topological neighbors of a particle. At each time step, by using the individual best position, and global best position, a new velocity for particle is updated. The computation for PSO is easy and adds only a slight computational load when it is incorporated into the conventional GA. Furthermore, the flexibility of PSO to control the balance between local and global exploration of the problem space helps to overcome premature convergence of elite strategy in GA, and also enhances search ability.

7.2.2 Genetic Algorithm with Euclidean Data Distance

When individuals in a genetic algorithm are differentiated to search for optimal solutions, there is a high chance for obtaining local optimal solutions. Using the conventional GA or PSO approach, optimal solutions are obtained mostly with some initial differentiated data and there is a high possibility for obtaining local optimal solutions. The proposed approach uses data points with the longest Euclidean distance for crossover process to avoid such local optimization. The idea is to obtain global solutions by considering the entire search space (all the data points). We consider the Euclidean distance for the function

$$F_1(x) = \sum_{i=1}^{2} x_i^2 \tag{7.1}$$

with the initial conditions as depicted in Table 7.1.

Figures 7.1 and 7.2 illustrate the relationship between objective function and the number of generations by a GA. Table 7.1 illustrates the initial conditions and Table 7.2 shows the Euclidean distance when applied to (7.1) and the relationship between the optimal value, average value, maximum and minimum values of the objective function.

As per proposed method, all the data points have a higher chance to be included in the search and thus a local solution could be avoided. The distance between two points on n search space is defined by

$$\text{distance} = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}. \tag{7.2}$$

To further demonstrate the performance, the Himmelblau function $F_2(x)$ is used:

$$F_2(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2. \tag{7.3}$$

The contour to obtain optimal solution by crossover using a conventional GA is illustrated in Fig. 7.3. Data points are selected by

Table 7.1. Initial conditions for the performance test

Function	Definition		No. of individuals	No. of iterations
	x_i^L	x_i^U		
$F_1(x) = \sum_{i=1}^{2} x_i^2$	−5.12	5.11	60	100

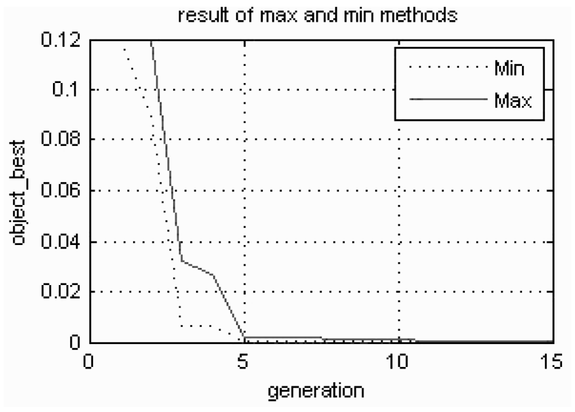

Fig. 7.1. Performance illustrating optimal objective function and generations for the test function F_1 using GA

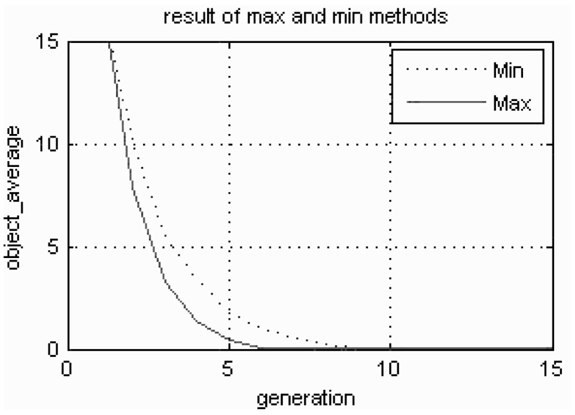

Fig. 7.2. Performance illustrating average objective function and number of generations for the test function F_1 using GA

Table 7.2. Performance results using the Min–Max method

	x_1	x_2	Optimal value of objective function	Average value of objective function
Max	$1.0885e-009$	7.1709e−010	1.6991e−018	3.5601e−013
Min	−2.2190e−011	$1.0253e-009$	1.0518e−018	3.7901e−013

$$A(x_1,y_1) \oplus B(x_1,y_1) \Rightarrow A',B'\left(x_1'|_{\min(x_1,y_1)}^{\max(x_1,y_1)},y_1'|_{\min(x_1,y_1)}^{\max(x_1,y_1)}\right) \tag{7.4}$$

As evident from Fig. 7.3, there is an optimal solution in only one place and optimal solution is obtained after 200 generations. Contour plot (Fig. 7.4) obtained

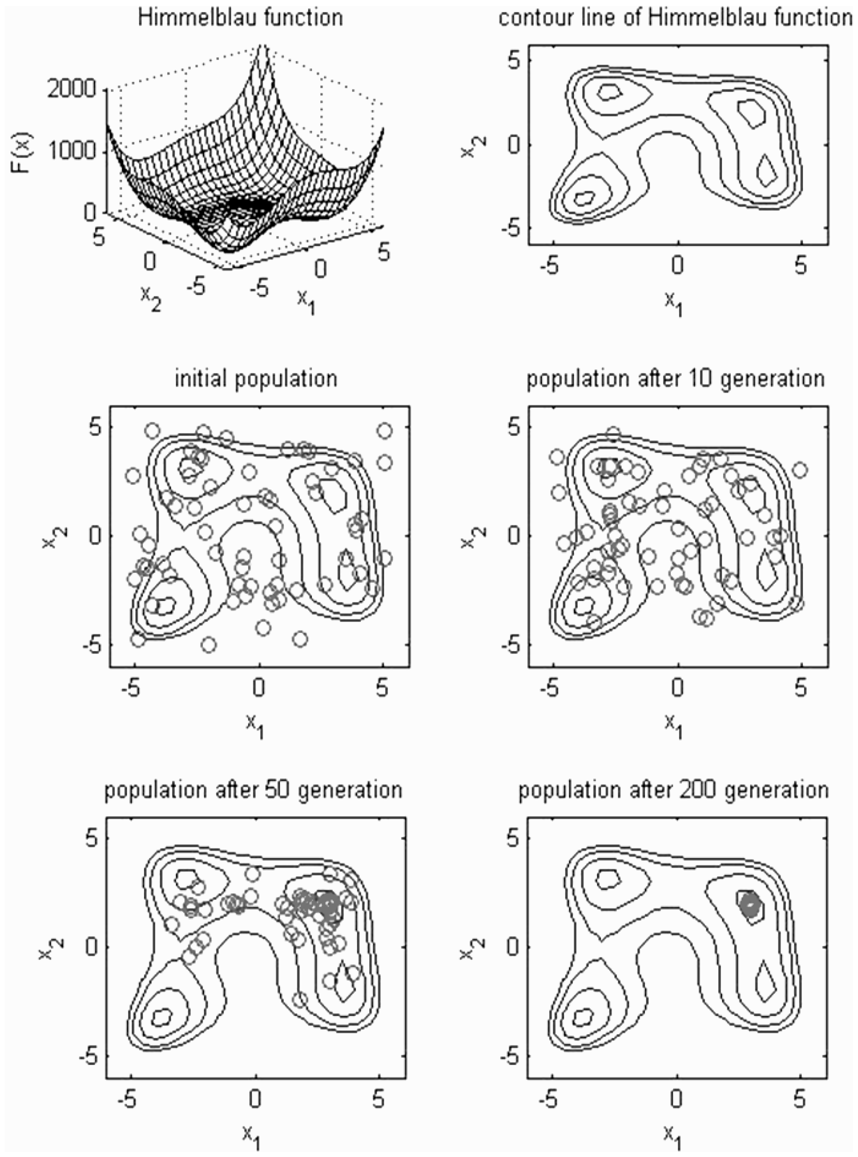

Fig. 7.3. Contour plot showing solutions for F_2 using Euclidean distance-based genetic algorithm

by the proposed genetic algorithm based on Euclidean distance shows that there are optimal solutions in both local and global spaces and its solution is obtained after 50 generations.

The influence of mutation in GA or in a hybrid system of PSO and GA has been studied to speed up the running time to obtain optimal solutions [12]. We used the

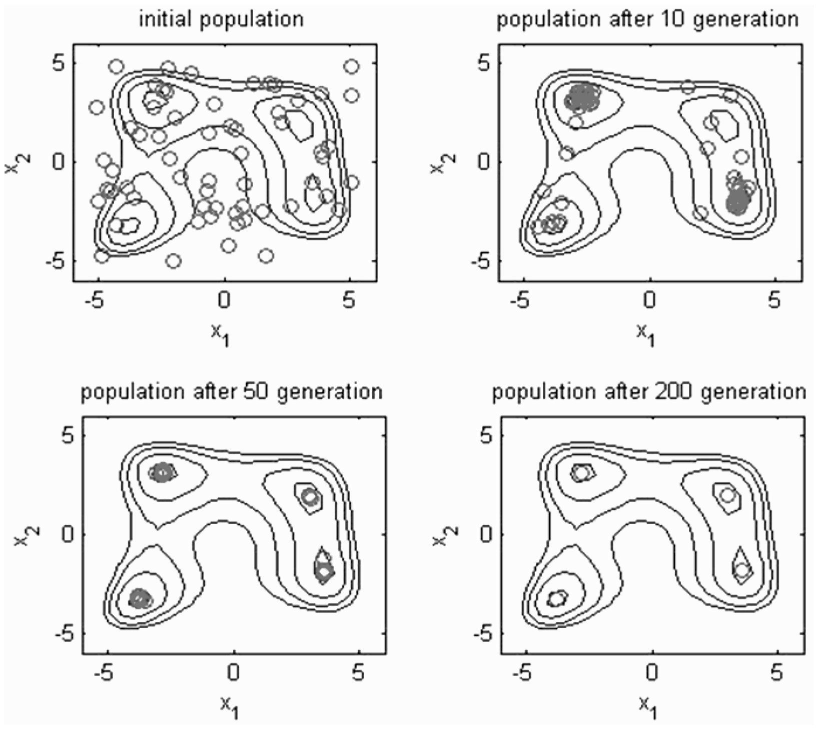

Fig. 7.4. Contour plot showing solutions for F_2 using genetic algorithm

position and speed vector of PSO as follows:

$$
\begin{aligned}
v_{f,g}^{(t+1)} &= wv_j^{(t)} + c_1^*\text{rand}()^*\left(p\text{best}_{j,g} - k_{j,g}^{(t)}\right) + c_2^*\text{Rand}()^*\left(g\text{best}_g - k_{j,k}^{(t)}\right) \\
& j = 1,2,\ldots,n \quad g = 1,2,\cdots,m \\
k_{j,g}^{(t+1)} &= k_{j,g}^{(t)} + v_{j,g}^{(t+1)}, k_g^{\min} \le k_{j,g}^{(t+1)} \le k_g^{\max}
\end{aligned}
\tag{7.5}
$$

where n is the number of agents in each group; m the number of members in each group; t the number of reproduction steps; $v_{j,g}^{(t)}$ the speed vector of agent j in reproduction step of tth, $V_g^{\min} \le V_{j,g}^{(t)} \le V_g^{\max}$ $k_{j,g}^{(t)}$ the position vector of agent j in reproduction step of tth; w the weighting factor; c_1, c_2 the acceleration constant; rand(), Rand() the random value between 0 and 1; $p\text{best}_j$ the optimal position vector of agent j; and $g$best is the optimal position vector of group.

The value of position vector and speed vector is determined by the acceleration constants c_1, c_2 . If these values are large, each agent moves to the target position with high speed and abrupt variation. If vice versa, agents wander about target place. As weighting factor w is for the search balance of the agent, the value for optimal search is given by

$$
w = w_{\max} - \frac{w_{\max} - w_{\min}}{\text{iter}_{\max}} \times \text{iter}, \tag{7.6}
$$

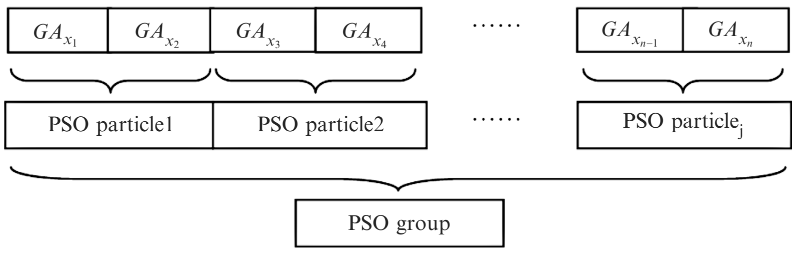

Fig. 7.5. Individual structure combined by PSO and GA

where $W_{\max}$ is the maximum value of $W(0.9)$; $W_{\min}$ the minimum value of $W(0.4)$; iter$_{\max}$ the number of iterations; and iter is the number of iterations at present.

The speed vector is limited by $V_g^{\min} \leq V_{j,g}^{(t)} \leq V_g^{\max}$. In this research, the value of speed vector for each agent is limited to 1/2 to avoid abrupt variation of position vector. Calculation process for each step is given in Fig. 7.5

[Step 1] Initialize all GA variables.
[Step 2] Initialize all PSO variables.
[Step 3] Calculate affinity of each agent for condition of optimal solution of GA. At this point, optimal position condition of PSO is introduced into the GA loop.
[Step 4] Arrange the group of PSO and agents in GA as shown in Fig. 7.6.
[Step 5] Update position vector *pbest* and speed vector *gbest*.
[Step 6] Perform crossover in GA using Euclidian distance and position vector of PSO.
[Step 7] Perform mutation in GA.
[Step 8] If condition of GA is satisfied with the target condition (iteration number or target value), reproduction procedure is halted. Otherwise, it goes to step 3. In Fig. 7.5, IG, ED, PV, and SV refers to initial group, Euclidean distance, position vector, and speed vector, respectively.

In this paper, initially, position of individuals are calculated by the Euclidean distance-based method and then mutation and crossover are performed to improve the running speed and to obtain global optimal solutions.

7.3 Experiment Results

7.3.1 Performance Analysis for Different Particle Sizes

To prove the learning structure suggested in this paper, function

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

is used as an example. Figure 7.7 illustrates the contour characteristics of the function.

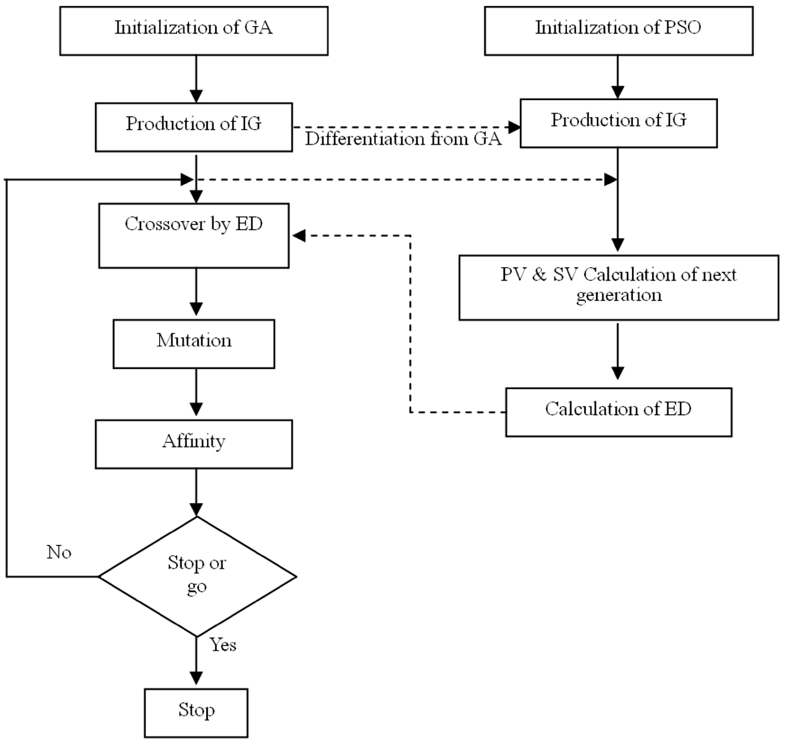

Fig. 7.6. Flowchart of GA–PSO algorithm

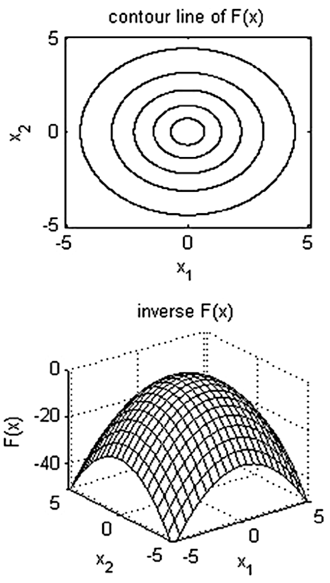

Fig. 7.7. Contour of function F_1

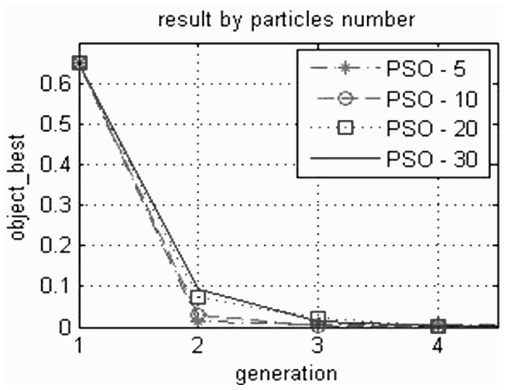

Fig. 7.8. Relationship between the objective function and number generations using different PSO settings

Figure 7.8 represents the relationship between objective function and number of GA generations for the number of particles in PSO and Fig. 7.9 illustrates the characteristics between the existing GA and the proposed EU–GA–PSO approach. As evident, the GA–PSO converges much faster than the conventional GA approach.

Table 7.3 depicts the relationship between variation of function and differentiation rate of PSO. When the differentiation rate is smaller, the convergence speed is faster but at the final step, the differentiation rate is larger and the convergent speed is faster.

7.3.2 Performance Characteristics of Hybrid GA–PSO Algorithm

Figure 7.10 illustrates the performance between GA and Hybrid GA–PSO. For comparison of both systems, test function, $F_1(x)$ and Euclidean data distance are used. Particle size for comparing the different characteristics is selected as 10. As evident from Fig. 7.10, after the first step, the conventional GA has faster convergence speed but during the final stages, GA–PSO has more stable speed because GA–PSO searches for optimal solution by incorporating position and direction for search (Tables 7.4 and 7.5).

7.3.3 Importance of GA Parameter Selection

In GA, in order to transfer gene information of parents or grandparents to offsprings effectively, differentiation is carried out through different selection schemes namely *RemSel* (Remainder stochastic Sample with Replacement Selection), *UnivSel* (Stochastic Universal Sampling Selection), and *RwSel* (Roulette Wheel Selection). Performance results are illustrated in Figs. 7.10–7.17 and initial condition of the considered four test functions are presented in Table 7.6.

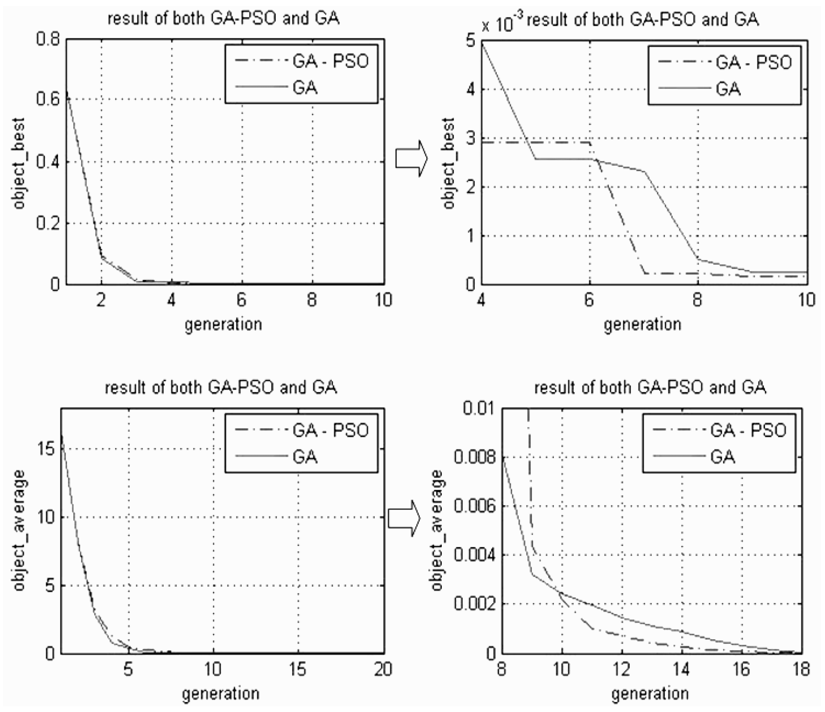

Fig. 7.9. Comparison between the conventional GA and GA–PSO

Table 7.3. Performance for different particle sizes

Particle size	$x_1(1.0\text{e}-006^*)$	$x_2(1.0\text{e}-006^*)$	The value of optimal objective function $(1.0\text{e}-012^*)$	The value of average objective function $(1.0\text{e}-008^*)$
5	0.3105	−0.4933	0.3398	0.0067
10	−0.2799	−0.1014	0.0886	0.1438
20	0.1655	0.3842	0.0550	0.0225
30	0.0325	0.0197	0.0014	0.0070

In Table 7.6, No. of IDs refer to the number of individuals and No. of Re; the number of reproduction steps, respectively.

For detailed illustrations, we used the following four test functions:

(1) Square function

$$F_1 = \sum_{1}^{3} x_i^2 .$$

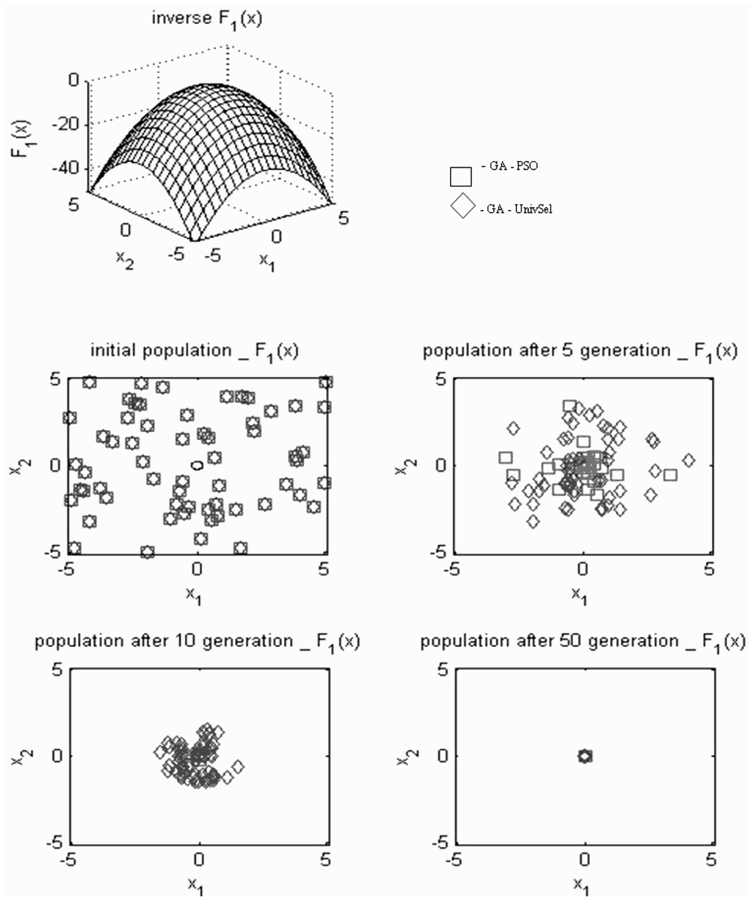

Fig. 7.10. Comparison between the conventional GA and GA–PSO for F_1

Table 7.4. Performance comparison for hybrid GA–PSO algorithm

	$x_1(1.0\text{e}-006^*)$	$x_2(1.0\text{e}-006^*)$	The value of optimal objective function	The value of average objective function
GA–PSO	0.0325	0.0197	1.4408e − 015	0.0700
GA	−0.2249	0.2585	1.1741e − 013	0.1962

Figures 7.10 and 7.11 depict the performance comparison between the conventional GA and the proposed GA–PSO approaches.

(2) Rosenbrock function

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

Figures 7.12 and 7.13 illustrate how the optimal solutions are obtained for the Rosenbrock function and Table 7.7 depicts the empirical results. As evident, GA–PSO has better convergence in the search for optimal solutions.

Table 7.5. Performance comparison for $F_1(x)$

	$x_1(1.0e-005^*)$	$x_2(1.0e-005^*)$	Optimal value of objective function	Average value of objective function $(1.0e-009^*)$
GA–PSO	-0.0001	-0.0001	$2.0656e-018$	$3.7940e-014$
GA-RemSel	-0.8788	-0.0064	$7.7228e-011$	$3.3378e-008$
GA-UnivSel	-0.3056	0.1973	$1.0706e-010$	$2.2349e-008$
GA-RwSel	0.3535	-0.9724	$1.3233e-011$	$5.3544e-008$

Table 7.6. Search space of test functions and initial conditions

Function	Definition		No.	No.	Particle
	x_i^L	x_i^U	of IDs	of Re	size
$F_1(x) = \sum_{i=1}^{2} x_i^2$	-5.12	5.11	60	100	10
$F_2(x) = 100(x_1^2 - x_2)^2 + (1-x_1)^2$	-2.048	2.047			
$F_3(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$	-6	6			
$F_4(x) = (0.002 + \sum_{j=1}^{25} (j + \sum_{i=1}^{2} (x_i - a_{ij})^6)^{-1})^{-1}$	-65.536	65.535			

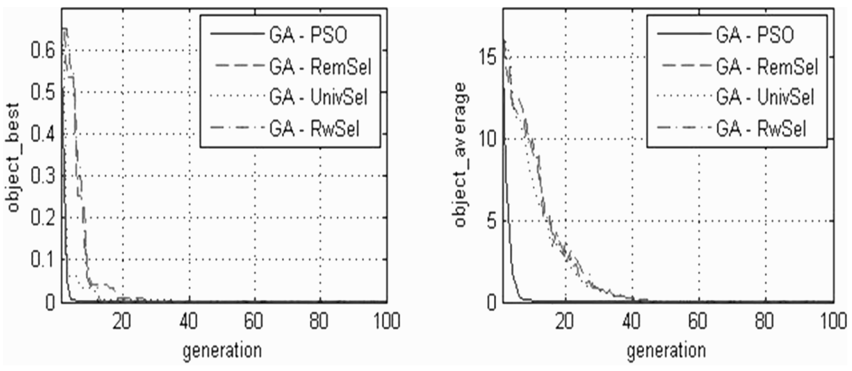

Fig. 7.11. Comparison between different selection schemes for F_1

(3) Himmelblau function

$$F_3(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

Figures 7.14 and 7.15 depict how the proposed method could fasten the convergence for the Himmelblau function. The GA–PSO method depicts better optimal solutions after 5 generations. On the other hand, after 50 generations, GA–PSO represents both optimal solutions (local optimal and global optimal)

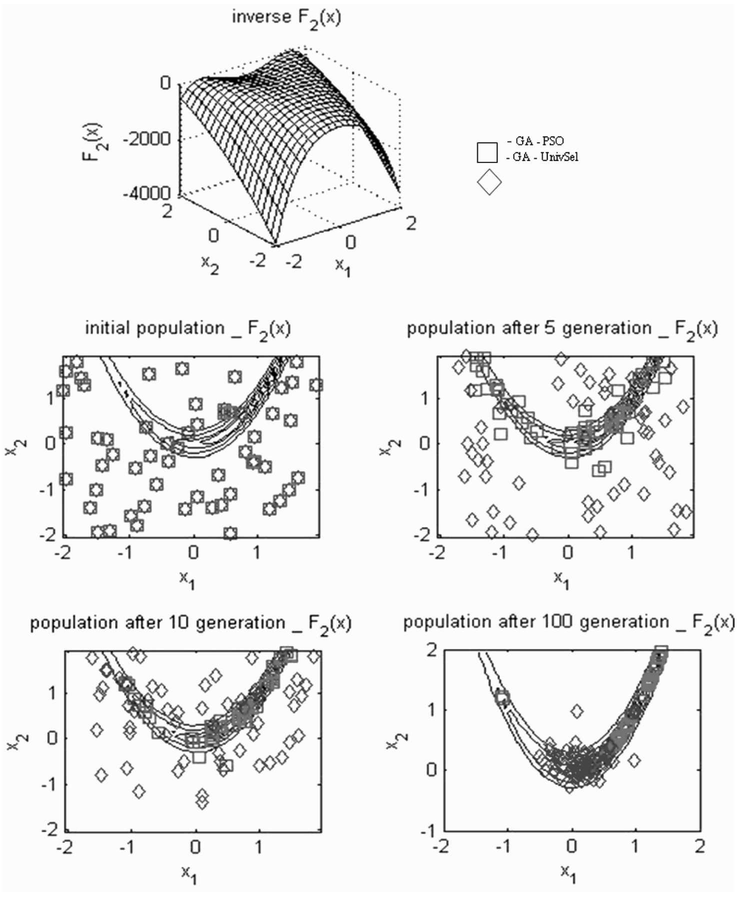

Fig. 7.12. Comparison between the conventional GA and GA–PSO for F_2

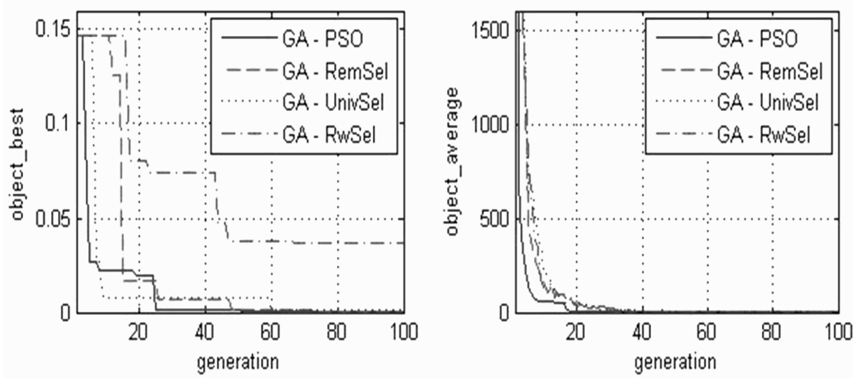

Fig. 7.13. Comparison between different selection schemes for F_2

Table 7.7. Performance Comparison for $F_2(x)$ using different selection schemes

	x_1	x_2	Optimal value of objective function	Average value of objective function
GA–PSO	1.0026	1.0052	6.7405e − 006	2.0807
GA-RemSel	0.9720	0.9447	7.8523e − 004	3.0355
GA-UnivSel	0.9612	0.9243	0.0015	5.4145
GA-RwSel	0.8084	0.6540	0.0367	1.2021

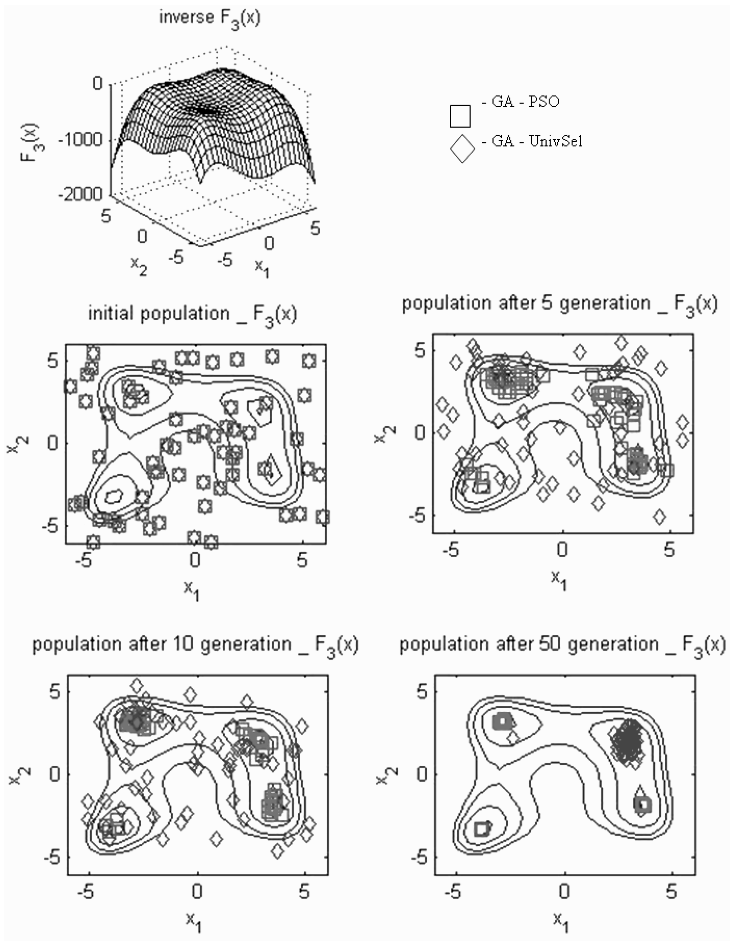

Fig. 7.14. Comparison between the conventional GA and GA–PSO for F_3

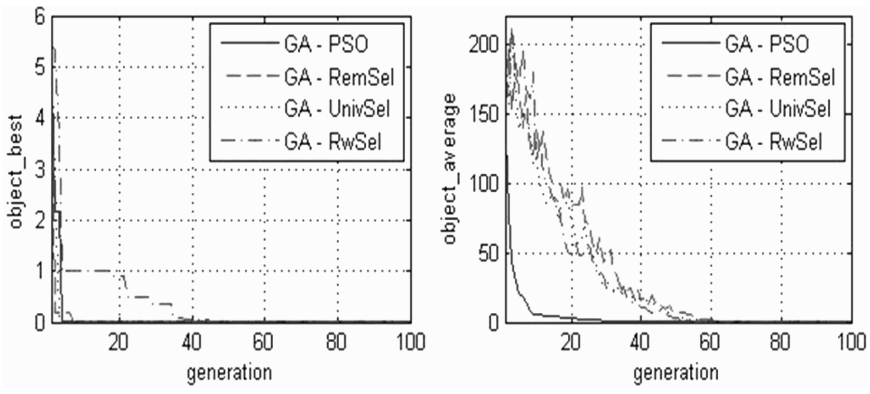

Fig. 7.15. Comparison between different selection schemes for F_3

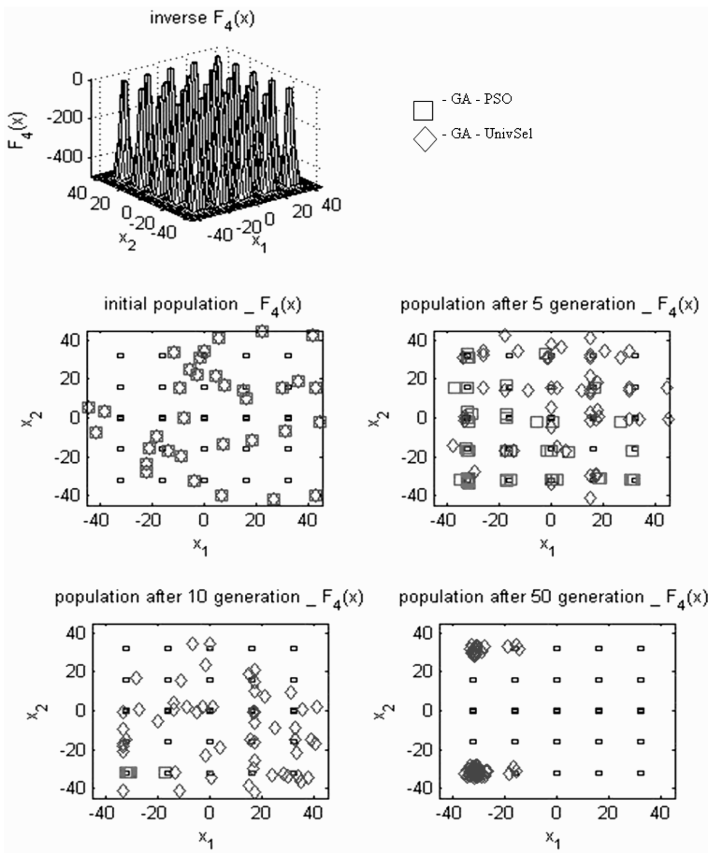

Fig. 7.16. Comparison between the conventional GA and GA–PSO for F_4

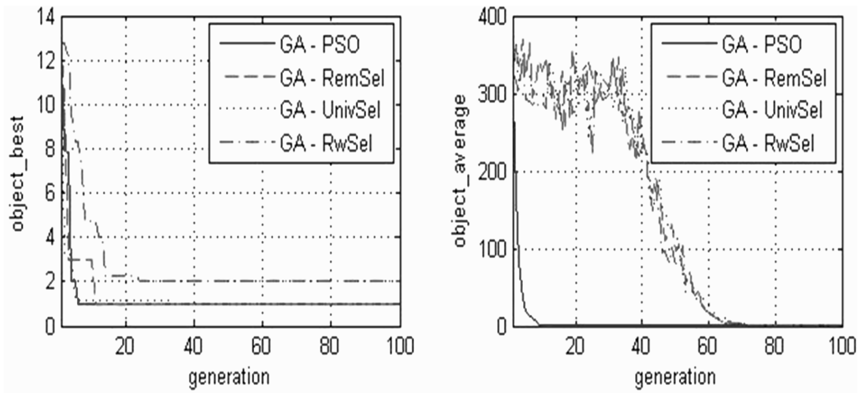

Fig. 7.17. Comparison between different selection schemes for F_4

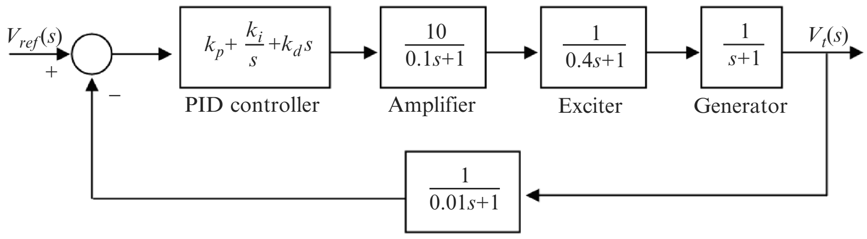

Fig. 7.18. Block diagram of an AVR system with a PID controller

but it reveals that it is possible to have a local optimal solution because the conventional method has optimal solution at one location.

(4) Fox hole function

$$F_4(x) = \left(0.002 + \sum_{j=1}^{25}\left(j + \sum_{i=1}^{2}(x_i - a_{ij})^6\right)^{-1}\right)^{-1}.$$

Figures 7.16 and 7.17 illustrate the performance results for the Fox hole function.

7.4 PID Controller Tuning for the AVR System

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

$$G(s) = k_p + \frac{k_i}{s} + k_d s \tag{7.7}$$

and block diagram of the AVR system is shown in Fig. 7.18. The performance index of control response is defined by

$$\begin{aligned}\min F(k_p,k_i,k_d) &= \frac{e^{-\beta}t_s/\max(t)}{\left(1-e^{-\beta}\right)|1-t_r/\max(t)|} + e^{-\beta}\text{Mo} + \text{ess} \\ &= \frac{e^{-\beta}(t_s+\alpha_2|1-t_r/\max(t)\text{Mo}|)}{(1-e^{-\beta}|1-t_r/\max(t)|)} + \text{ess} \\ &= \frac{e^{-\beta}\left(t_s/\max(t)+\alpha\text{Mo}\right)}{\alpha} + \text{ess}\end{aligned} \tag{7.8}$$

where k_p, k_i, k_d is the parameters of PID controller; β the weighting factor; Mo the overshoot; t_s the settling time (2%); ess the steady-state error and t the desired settling time.

In (7.8), if the weighting factor, β increases, rising time of response curve is small, and when β decreases, rising time is big.

Performance criterion is defined as Mo $= 50.61\%$, ess $= 0.0909$, $t_r = 0.2693(s)$, $t_s = 6.9834(s)$ and the following parameter settings were used.

In PSO, the number of each agent is fixed as 10 with the number of groups as 5:

Weighting factor: $w_{\max} = 0.9, w_{\min} = 0.4$
Restriction of velocity vector:

$$V_{k_p}^{\max} = K_p^{\max}/2, V_{k_i}^{\max} = K_i^{\max}/2, V_{k_d}^{\max} = K_d^{\max}/2, V_{k_p,k_i,k_d}^{\min} = -V_{k_p,k_i,k_d}^{\max}$$

Acceleration constant:$c_1 = 2, c_2 = 2$.

Terminal voltage step response of an AVR system without controller is given in Fig. 7.19 and the characteristics of the AVR system obtained due to the variation of β using GA, PSO, and hybrid EU–GA–PSO approach is illustrated in Figs. 7.20–7.32. Empirical results are summarized in Tables 7.8–7.9.

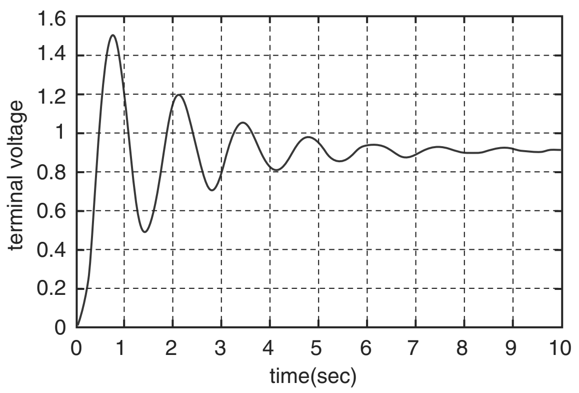

Fig. 7.19. Terminal voltage step response of an AVR system without controller

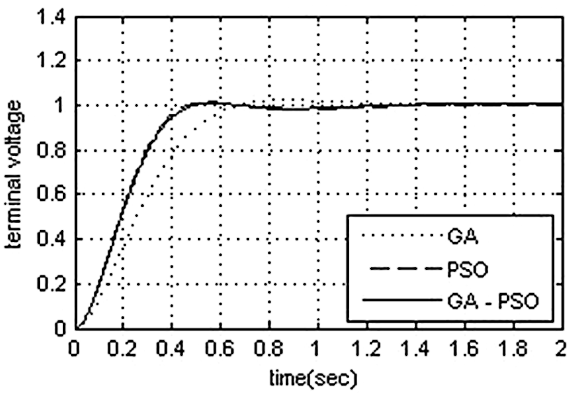

Fig. 7.20. Terminal voltage step response of an AVR system with a PID controller ($\beta = 0.5$, generations = 50)

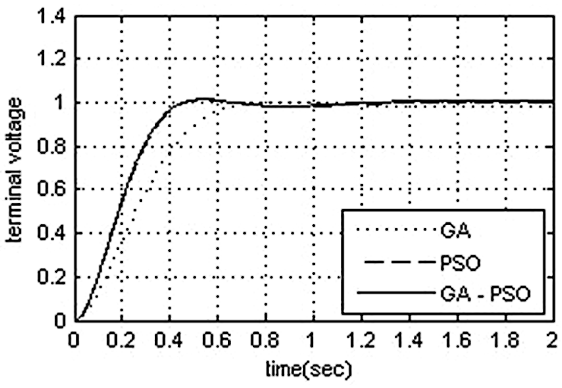

Fig. 7.21. Terminal voltage step response of an AVR system with a PID controller ($\beta = 0.5$, generations = 100)

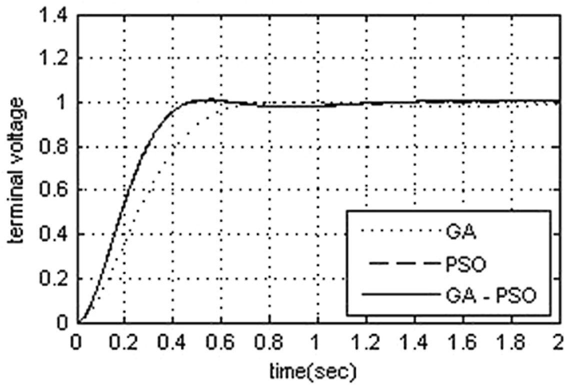

Fig. 7.22. Terminal voltage step response of an AVR system with a PID controller ($\beta = 0.5$, generations = 150)

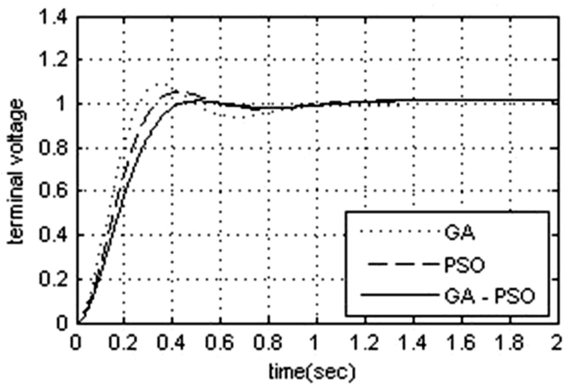

Fig. 7.23. Terminal voltage step response of an AVR system with a PID controller ($\beta = 1.0$, generations = 10)

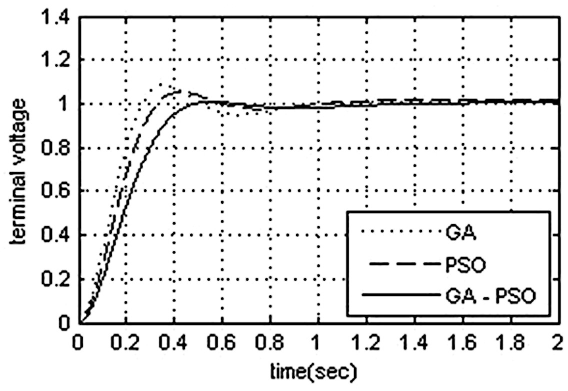

Fig. 7.24. Terminal voltage step response of an AVR system with a PID controller ($\beta 1.0$, generations = 50)

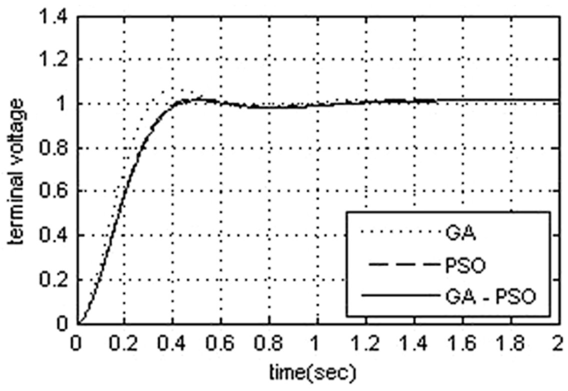

Fig. 7.25. Terminal voltage step response of an AVR system with a PID controller ($\beta = 1.0$, generations = 100)

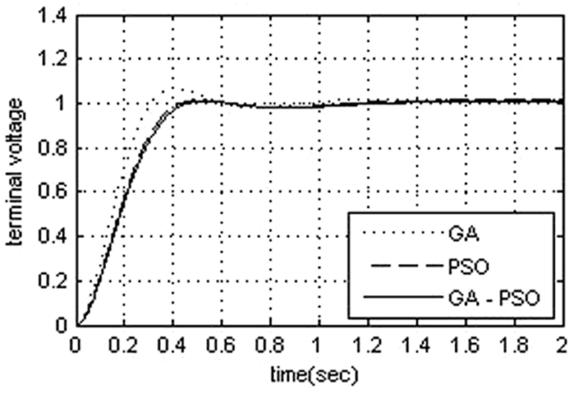

Fig. 7.26. Terminal voltage step response of an AVR system with a PID controller (β=1.0, generations=150)

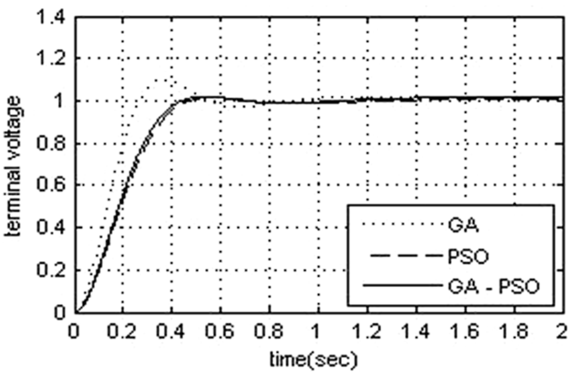

Fig. 7.27. Terminal voltage step response of an AVR system with a PID controller (β=1.5, generations=10)

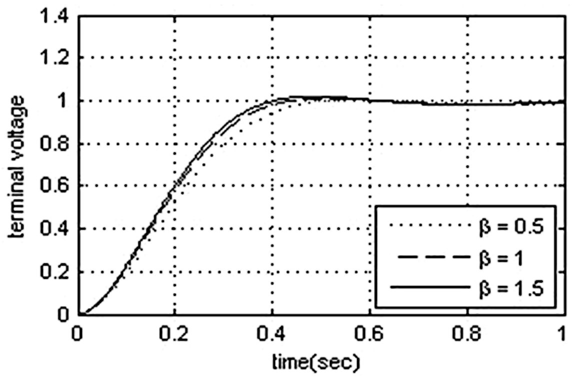

Fig. 7.28. Terminal voltage step response of an AVR system with a PID controller for different β values and for 25 generations

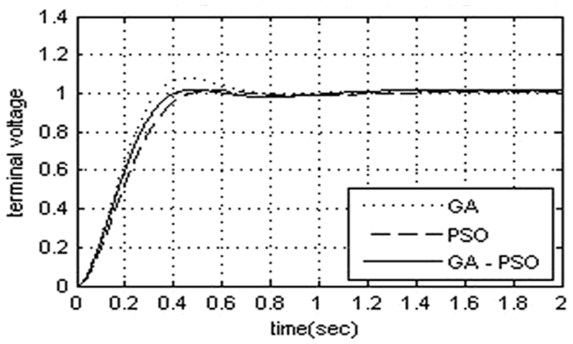

Fig. 7.29. Terminal voltage step response of an AVR system with a PID controller (β=1.5, generations=50)

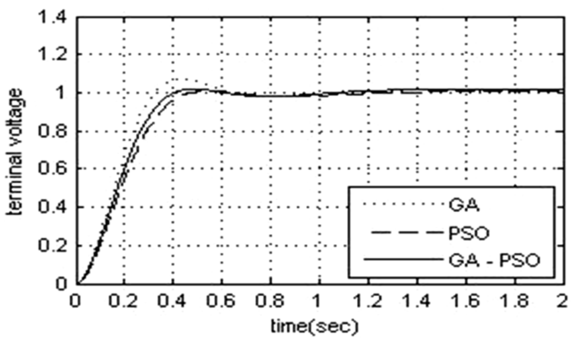

Fig. 7.30. Terminal voltage step response of an AVR system with a PID controller (β=1.5, generations= 100)

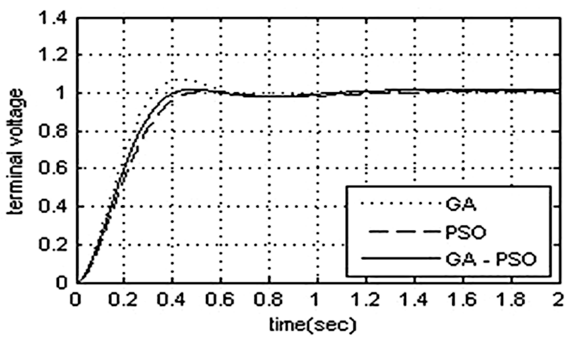

Fig. 7.31. Terminal voltage step response of an AVR system with a PID controller (β=1.5, generations= 150)

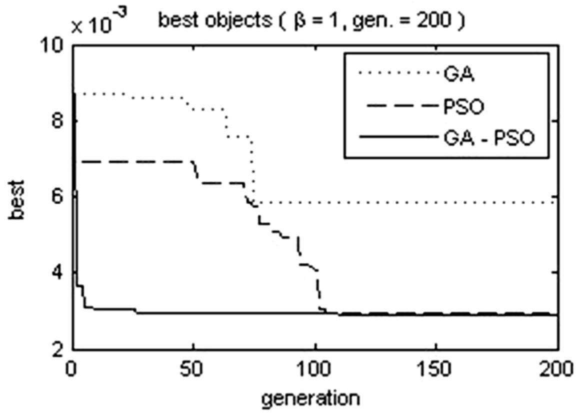

Fig. 7.32. Comparison of the best objective values using all the methods ($\beta = 1.5$, generations $= 200$

Table 7.8. Simulation results of PID controller in AVR system for β variation

β	Number of generations	k_p	k_i	k_d	Mo(%)	ess	t_s	t_r	Objective
0.5	25	0.6204	0.4929	0.2232	0.97	0.0097	0.4570	0.2973	0.0079
1	25	0.6584	0.5819	0.2548	1.71	0.0166	0.4000	0.2651	0.0030
1.5	25	0.6801	0.6260	0.2681	1.97	0.0186	0.3770	0.2523	0.0072

7.5 Conclusions

In this chapter, a hybrid combination of Euclidean distance-based genetic algorithm and PSO algorithm are introduced for function optimization and the enhancement of optimal tuning of the conventional PID controller. By incorporating the Euclidean distance measure for selecting mutation or crossover points, the search space is well explored. Therefore, GA can provide exact optimal solution, while it can avoid local optimal solutions. Experiment results reveal the efficiency of the proposed approach with a faster convergence and optimal solutions. The GA–PSO system proposed in this chapter could be easily extended to model other complex problems involving local optimal and global optimal solutions.

Owing to their popularity in the industrial world, over the past 50 years, several approaches for determining PID controller parameters have been developed for stable processes that are suitable for autotuning and adaptive control and for single input single output (SISO) systems. In spite of the enormous amount of research work reported in the tuning approaches, many PID controllers are poorly tuned in practice. One of the reasons is that most of the tuning methods are derived for particular processes and situations, and therefore apply well only to their own areas. It is a

Table 7.9. Simulation results for PID controller tuning using different methods

β	No. of generations	Controller type	k_p	k_i	k_d	Mo(%)	ess	t_s	t_r	Evaluation value
0.5	50	GA	0.5045	0.3654	0.1400	2.3004	0.0005	1.0010	0.4136	0.0176
		PSO	0.6572	0.4816	0.2284	1.4589	0.0035	0.4280	0.2842	0.0077
		GA–PSO	0.6310	0.4929	0.2232	0.8857	0.0083	0.4480	0.2941	0.0077
	100	GA	0.5045	0.3080	0.1400	0.2261	0.0148	0.6590	0.4231	0.0108
		PSO	0.6572	0.4816	0.2284	1.4589	0.0035	0.4280	0.2842	0.0077
		GA–PSO	0.6554	0.5224	0.2374	1.3606	0.0095	0.4190	0.2779	0.0075
	150	GA	0.5045	0.3142	0.1416	0.2254	0.0128	0.6570	0.4214	0.0108
		PSO	0.6537	0.4852	0.2292	1.3115	0.0045	0.4300	0.2845	0.0076
		GA–PSO	0.6447	0.5060	0.2350	0.8581	0.0086	0.4300	0.2825	0.0073
	200	GA	0.5061	0.3080	0.1420	0.0044	0.0149	0.6600	0.4211	0.0106
		PSO	0.6491	0.4915	0.2317	1.0493	0.0059	0.4300	0.2839	0.0075
		GA–PSO	0.6447	0.5058	0.2350	0.8564	0.0085	0.4300	0.2825	0.0073
1	50	GA	0.9186	0.8100	0.3935	8.7179	0.0122	0.8550	0.1758	0.0083
		PSO	0.7893	0.7196	0.3105	5.2491	0.0154	0.8320	0.2155	0.0069
		GA–PSO	0.6415	0.4825	0.2274	0.8820	0.0055	0.4400	0.2889	0.0030
	100	GA	0.8326	0.8100	0.3277	6.8331	0.0171	0.5570	0.2037	0.0058
		PSO	0.6834	0.6096	0.2611	2.2200	0.0164	0.5340	0.2559	0.0040
		GA–PSO	0.6657	0.5697	0.2548	1.4503	0.0143	0.3980	0.2639	0.0029
	150	GA	0.8326	0.8100	0.3277	6.8331	0.0171	0.5570	0.2037	0.0058
		PSO	0.6651	0.5690	0.2533	1.4511	0.0142	0.3990	0.2649	0.0029
		GA–PSO	0.6523	0.5189	0.2398	1.0510	0.0094	0.4200	0.2773	0.0029
	200	GA	0.8326	0.8100	0.3277	6.8329	0.0171	0.5570	0.2037	0.0058
		PSO	0.6660	0.5682	0.2543	1.4285	0.0140	0.3980	0.2641	0.0029
		GA–PSO	0.6522	0.5188	0.2398	1.0472	0.0094	0.8680	0.2773	0.0029
1.5	50	GA	0.8486	0.7165	0.2817	8.1337	0.0088	0.6690	0.2214	0.0155
		PSO	0.6473	0.5000	0.2245	1.5674	0.0072	0.4350	0.2885	0.0079
		GA–PSO	0.6801	0.6227	0.2681	1.9368	0.0183	0.3780	0.2523	0.0072
	100	GA	0.8365	0.6903	0.3010	6.8031	0.0090	0.5920	0.2150	0.0136
		PSO	0.6446	0.5050	0.2343	0.8779	0.0084	0.4300	0.2830	0.0074
		GA–PSO	0.6795	0.6177	0.2681	1.8665	0.0179	0.3780	0.2525	0.0071
	150	GA	0.8283	0.7143	0.3010	6.7151	0.0112	0.5950	0.2156	0.0135
		PSO	0.6446	0.5044	0.2345	0.8612	0.0083	0.4300	0.2829	0.0073
		GA–PSO	0.6795	0.6168	0.2681	1.8573	0.0178	0.3780	0.2526	0.0071
	200	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

common experience that we are not certain which tuning method should be chosen to provide good control to a given process. Intelligent controllers could even self-initialize and recalibrate even with little a priori knowledge especially due to the occurrence of significant changes in the process dynamics.

References

1. Matsummura S (1998), Adaptive control for the steam temperature of thermal power plants, Proceedings of the 1993 IEEE on Control Applications, pp. 1105–1109
2. Kim DH (2004), Robust PID controller tuning using multiobjective optimizaion based on clonal selection of immune algorithm, Proceedings of the International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Springer, Berlin Heidelberg New York, pp. 50–56
3. Lee CH, Ten CC (2003), Calculation of PID controller parameters by using a fuzzy neural network, ISA Transaction, pp. 391–400
4. Mann KI, Hu BG, and Gosine RG (1999), Analysis of direction fuzzy PID controller structures, IEEE Transactions on Systems, Man, and Cybernetics Part B, vol. 29, no. 3, pp. 371–388
5. Lin CL, Su HW (2000), Intelligent control theory in guidance and control system design: an Overview, Proceedings of the National Science Council ROC(A), vol. 24, no. 1, pp. 15–30
6. Fleming PJ and Purshouse RC (2002), Evolutionary algorithms in control system engineering: A survey, control engineering practice, vol. 10, pp. 1223–1241
7. Gaing ZL (2004), A particle swarm optimization approach for optimum design of PID controller in AVR system, IEEE Transactions on Energy Conversion vol. 19, no. 2, pp. 384–391
8. Eberchart R and Kennedy J (1995), A new optimizer using particle swarm theory, Proceedings of the International Symposium on Micro Machine and Human Science, pp. 39–43
9. Michalewicz Z (1999), Genetic algorithms+data structures=evolution programs. Springer, Berlin Heidelberg New York
10. Shi Y and Eberhart R (1998), A modified particle swarm optimizer, Proceedings of the IEEE World Congress on Computational Intelligence, pp. 69–73
11. Yoshida H, Kawata K, and Fukuyama Y (2000), A particle swarm optimization for reactive power and voltage control considering voltage security assessment, IEEE Transactions on Power Systems, vol. 15, pp. 1232–1239
12. Juang CF (2004), A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, Systems, Man and Cybernetics, Part B, IEEE Trans. vol. 34 (2), pp. 997–1006