Control of nonlinear systems using a hybrid APSO-BFO algorithm: An optimum design of PID controller

Alireza Alfi[1⊠], Mehdi Siahi[2]
(1)Faculty of Electrical and Robotic Engineering, Shahrood University of Technology, Shahrood, 36199-95161, Iran.
(2)Department of Electrical Engineering, Garmsar Branch, Islamic Azad University, Garmsar, 91775-1111, Iran.
a_alfi@shahroodut.ac.ir; mahdi_siahi@yahoo.com

Abstract

This paper proposes a novel hybrid algorithm namely APSO-BFO which combines merits of Bacterial Foraging Optimization (BFO) algorithm and Adaptive Particle Swarm Optimization (APSO) algorithm to determine the optimal PID parameters for control of nonlinear systems. To balance between exploration and exploitation, the proposed hybrid algorithm accomplishes global search over the whole search space through the APSO algorithm whereas the local search is performed by BFO algorithm. The proposed algorithm starts with APSO algorithm. In the proposed APSO, every particle dynamically adjusts inertia weight according to feedback taken from particles best memories. In this case, APSO algorithm is used to enhance global search ability and to increase convergence speed. When the change in fitness value is smaller than a predefined value, the searching process is switched to BFO to accelerate the search process and find an accurate solution. In this way, this hybrid algorithm may find an optimum solution more accurately. To demonstrate the effectiveness of the proposed algorithm, its results are compared with those obtained by Basic PSO (BPSO), Standard BFO (SBFO), BFO with PSO (PSO-BFO), BFO with GA (GA-BFO) and Differential Evolution with BFO (DE-BFO). The numerical simulations are shown the potential of proposed algorithm.

Keywords: Bacterial foraging optimization algorithm, Particle swarm optimization, PID controller, Genetic algorithm, Differential evolution

1. Introduction

Process control techniques in the industry have made great advances. Although various control approaches such as adaptive control and neural control have been studied during two past decades [1], but the most well-known controller is yet the proportional integral derivative (PID) controller [2-5].

PID controller has been widely utilized in the industry because of its simple realization and robust performance within a wide range of operating conditions. Unfortunately, it has been quite difficult to properly tune the gains of PID controller since many industrial processes are often burdened with problems such as nonlinearities and time delays. The first method utilized the classical tuning rules proposed by Ziegler and Nichols (ZN). In general, it is often hard to determine the optimal PID parameters with the ZN formula in various industrial processes. One of shortage on this method is the necessary of the prior knowledge regarding plant model. Based on this, it is highly

81

desirable to increase the capabilities of PID controllers by the proper tuning of the controller parameters. Hence, many artificial intelligence (AI) techniques such as fuzzy systems have been employed to improve the PID controller performances for a wide range of processes whilst retaining their basic characteristics. Beside these methods, several heuristic optimization techniques such as genetic algorithm (GA) and particle swarm optimization (PSO) algorithm seem to be a promising alternative to traditional techniques [6-9]. These algorithms are currently gaining popularity in solving engineering optimization problems.

Compared to GA, PSO takes less time for each function evaluation as it is free from the complex computation in genetic algorithm such as coding/decoding, crossover and mutation [10]. Because of the simple concept, easy implementation and rapidly converging towards an optimum, PSO has gained increasing popularity in recent years to solve effectively a plethora of problems in science and engineering [11-14].

Although PSO has shown some significant advances by providing high speed of convergence in specific problems, but it does exhibit some drawbacks. These drawbacks are summarized as follows:

1) It sometimes is easy to be trapped in local optima when facing to complex optimization problem. So it suffers from the premature convergence problem.
2) The convergence rate decreased significantly in the later period of evolution; while reaching a near optimal solution, the algorithm stops optimizing, and therefore the accuracy of algorithm is limited [15].

Motivated by the aforementioned before, this paper proposes a novel methodology by incorporating an adaptive particle swarm optimization (APSO) and bacterial foraging optimization (BFO) algorithm, namely APSO-BFO algorithm. The main goal of this paper is to illustrate that the integration of some features from both the PSO and the BFO, can prove very effective in tackling many nearly complex optimization problems on which both the basic algorithms execute poorly. Based on this, to overcome the first shortage, the APSO algorithm is utilized to enhance global search ability of PSO. Moreover, to avoid the second shortage of PSO, the proposed APSO is combined with BFO. To show the feasibility of the proposed APSO-BFO, in the beginning it is extensively compared with Basic PSO (BPSO), Standard BFO (SBFO), BFO with PSO (PSO-BFO), BFO with GA (GA-BFO) and Differential Evolution with BFO (DE-BFO) over a test suit of five well-known benchmark functions. Results depict that the proposed algorithm has better performance than others. Next, APSO-BFO algorithm is employed to determine optimal tuning of the PID controller parameters for control of nonlinear systems.

The remaining content of this paper is organized as follows: Section 2 introduces the BPSO and BFO algorithms shortly. In Section 3, the proposed algorithm referred to as APSO-BFO algorithm is described. The superiority of the proposed algorithm both in robustness and efficiency is verified over a few numerical benchmarks and a practical PID tuner design problem in section 4 and 5, respectively. Moreover, numerical simulations and comparisons are provided in these sections. Finally, section 6 summarizes and draws conclusions.

2. Preliminaries

In this section, we briefly describe both BPSO and SBFO algorithms.

2.1 BPSO algorithm

PSO is a very popular optimization algorithm these days and it draws inspiration from the group behavior of a bird flock. In PSO, multiple candidate solutions coexist and collaborate simultaneously. Each solution is referred as a 'particle'. Each particle flies in the search space looking for the optimal position. As time passes through its quest, each particle adjusts its position according to its own experience and the experience of neighboring particles.

In BPSO, the velocity and position updating rule is given by (Kennedy et al., 2001):

$$v_{id}^{t+1} = v_{id}^t + c_1 r_1 (pbest_{id}^t - x_{id}^t) + c_2 r_2 (gbest_d^t - x_{id}^t) \tag{1}$$

$$x_{id}^{t+1} = x_{id}^t + v_{id}^{t+1}; \quad i = 1,2...,n \tag{2}$$

where c_1 and c_2 are constants named acceleration coefficients, respectively, r_1 and r_2 are two independent random numbers uniformly distributed in the range of [0, 1], $V_i \in [-V_{min}, V_{max}]$, where V_{max} is a problem-dependent constant defined in order to clamp the excessive roaming of particles and $pbest_{id}^t$ is the best previous position along the dth dimension of ith particle in iteration t (memorized by every particle). Finally, $gbest_{id}^t$ is the best previous position among all the particles along the dth dimension in iteration t (memorized in a common repository).

2.2 SBFO algorithm

The BFO algorithm was first proposed by Passino [16] based on the search and optimal foraging strategies of the *E. coli* bacteria. Nowadays BFO has been successfully utilized in some optimal problems such as harmonic estimation [17], machine learning [18] and transmission loss reduction [19]. The idea in this algorithm was adopted from biological and physical living behavior of *E. coli* bacteria existing in human intestine. Chemotaxis is basically a behavior to earn a living that performs a type of optimization in which bacteria try to reach the nutrients and avoid noxious materials and find a way to exit the neutral and noxious nutrient environment. The bacterial swarm proceeds through four steps namely chemotaxis, swarming, reproduction and elimination-dispersal. A brief description of each of these processes will be described. A detailed description can be traced in [16].

2.2.1 Chemotaxis

This step simulates the movement of an *E. coli* bacterium in BFO algorithm. An *E. coli* can move in two different ways: It can swim for a period of time in the same direction or it may tumble, and alternate between these two modes of operation for the entire lifetime. The following equation represents this movement.

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i) \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{3}$$

where $\theta^i(j,k,l)$ indicated the position of ith bacterium at jth chemotaxis, kth reproduction, and lth elimination and dispersal, respectively. Moreover, $C(i)$ and $\Delta(i)$ are the movement vector length and direction random vector, respectively.

2.2.2 Swarming

The explanation of Chemotaxis step is for the cases while bacteria behaved separately, i.e. without producing signal for other bacteria, however there is an exchange of signals between the bacteria here (through absorbing materials). Hence, the group movement for all bacteria is defined as follows:

$$
J_{cc}(\theta_{gm}(j,k,l),\theta(j,k,l)) = \sum_{i=1}^{s} j_{cc}^{i}(\theta_{gm}(j,k,l),\theta^{i}(j,k,l))
$$

$$
= \sum_{i=1}^{s} j_{cc}^{i}(\theta_{gm}(j,k,l),\theta^{i}(j,k,l)) = \sum_{i=1}^{s}\left[-d_{attract}\exp\left(-\omega_{attract}\right)\sum_{m=1}^{P}(\theta_{m_{gm}}-\theta_{m}^{i})^2\right]+ \tag{4}
$$

$$
\sum_{i=1}^{s}\left[h_{repellant}\exp\left(-\omega_{repellant}\right)\sum_{m=1}^{P}(\theta_{m_{gm}}-\theta_{m}^{i})^2\right]
$$

where $J_{cc}(\theta(j,k,l))$ is the cost function value to be added to the actual cost function (to be minimized) to present a time varying cost function in jth chemotactic, kth reproduction, and lth elimination stage, θ_{gm} is the location of the global minimum bacterium. Moreover, $\theta_{m_{gm}}$ represents the mth parameter of the global minimum bacteria, S is the total number of bacteria, P is the number of variables to be optimized, which are present in every bacterium $\theta=[\theta_1,\dots\theta_P]^T$ is a point in the P dimensional search domain. Finally, $d_{attract}$, $h_{repellant}$, $\omega_{attract}$ and $\omega_{repellant}$ are different parameters that should be chosen appropriately.

2.2.3 Reproduction

After N_c chemotactic steps, the reproduction step is taken. Suppose that N_{re} be the number of reproduction steps and the number of population members denoted by S be a positive even integer number. The least healthy bacteria ($S_r = \frac{S}{2}$) ultimately die while each of the S_r healthier bacteria (those yielding higher value of fitness function) asexually split into two bacteria which are placed in the same location. So, the number of bacteria is always S. Using this strategy, the swarm size remains constant.

2.2.4 Elimination and Dispersal

Although swimming prepares the environment for local foraging and speeds up convergence in the reproduction process, but only by swimming and reproduction, a large space cannot be adequate for searching the global optimal solution. In BFO, the dispersal event occurs after a definite number of the reproduction processes. First, a bacterium with regard to a *Ped* prearranged probability is chosen to move and disperse to another position in the environment. These events can effectively prevent trapping in the local optima. Second, *Ped* is defined for each bacterium which is the probability of elimination and dispersal while *Ned* presents the number of elimination and dispersal event. Moreover, it is assume that the frequency of reproduction is more than the elimination and dispersal event and the frequency of the moving steps is more than the frequency of the reproduction steps. As a result, many regeneration steps take place before elimination and dispersion. Furthermore, many movement steps occur before reproduction [20].

3. The Proposed Hybrid APSO-BFO Algorithm

3.1 APSO Algorithm

Although PSO is an optimization algorithm that proves to be efficient, straightforward and robust, but if no care is taken the velocities perhaps attain large values, particularly when particles are far away from local and global bests. Inertia weight plays the role of balancing the global search and local search abilities of the swarm. Empirical studies of PSO with inertia weight have shown that a relatively large inertia weight have more global search ability while a relatively small inertia weight results in a faster convergence. Consequently, to eliminate this drawback, some empirical and theoretical studies were proposed [7][20]. Nevertheless these algorithms improve the performance of PSO, they can not truly reflect the actual search process without any feedback taken from how far particle's fitness are from the estimated (or real) optimal value, when the real optimal value is known in advance.

To overcome this drawback, Modares et. al [20] proposed a novel Adaptive PSO (APSO). In the proposed APSO algorithm, to incorporate the difference between particles into PSO, the inertia weight is variable with the number of particles. To truly reflect the actual search process, the inertia weight is also set according to feedback taken from particles best memories. Due to these reasons, in this algorithm, the inertia weight is dynamically adapted for every particle by considering a measure called Adjacency Index (AI), which characterizes the nearness of individual fitness to the real optimal solution. Based on this index, every particle could decide how to adjust the values of inertia weight. For this purpose, the velocity updating rules in the proposed APSO is given by

$$v_i^{t+1} = \omega_i^t v_i^t + c_{1i}^t r_1(pbest_i^t - x_i^t) + c_2^t r_2(gbest^t - x_i^t) \tag{5}$$

In Eq. (5), $\omega_i^t = \dfrac{1}{1+e^{-(\alpha \times AI_i^t)^{-1}}}$ where α is a positive constant in the range (0,1].

Compared with that in BPSO, the velocity updating given in Eq. (3) has two different characteristics:

1) To incorporate the difference between particles into BPSO, so that it can simulate a more precise biological model, the inertia weight is variable with the number of particles.

2) To truly reflect the actual search process, the inertia weight is set according to feedback taken from particles best memories.

During the search mechanism, the particles face different finesses; as a result they get different values of AI and then inertia weight. While the fitness of a particle is far away from the real global optimal, AI for this particle has a small value (a low adjacency) and the value of inertia weight will be large resulting strong global search abilities and locate the promising search areas. Meanwhile, the fitness of a particle achieves near the real global optimal, AI for this particle has a big value (a high adjacency) and inertia weight will be set small, depending on the nearness of its best fitness to the optimal value, to facilitate a finer local explorations and so accelerate convergence.

3.2 The Proposed Hybrid Algorithm

Based on above mentioned, BPSO may converges to local optimal solutions whereas BFO dependent on random direction which slows down the optimal solution process. Hence, in the proposed APSO-BFO algorithm, it is tried to benefit from the advantages of these two algorithms. In the process of optimization, APSO-BFO algorithm combines local search method (through self experience) with global search (through neighboring experience) method, attempting to balance exploration and exploitation. The proposed hybrid algorithm performs local search through the chemotactic movement operation of BFO whereas the APSO accomplishes the global search over the entire search space. Moreover, in APSO-BFO, after undergoing a chemo-tactic step, the APSO mutates every bacterium. In this stage, the bacterium is stochastically attracted towards the globally best position found so far in the whole population at current time and also towards its previous heading direction. The procedure for the proposed algorithm can be summarized as follows:

Step 1: Initialize parameters.

Step 2: Update the fitness value of the *i*th bacterium in the *j*th chemo-taxis and *k*th reproduction loop, the position vector of the best position found by all bacteria and the fitness of the best position found so far.

Step 3: Reproduction loop.

Step 4: Chemotaxis loop with mutation by APSO algorithm.

Step 5: The least healthy bacteria with highest fitness function values die and the other half of bacteria population with the best values split.

Step 6: If k is less than the number of reproduction steps, go to step 1.

Step 7: Elimination-dispersal.

4. Experimental and Evaluation Performance

To test the performance of the proposed APSO-BFO algorithm, five well-known benchmark functions is utilized as shown in table 1, and its results are compared to the performance of other algorithms such as BPSO, SBFO, GA-BFO [21], PSO-BFO [22], and DE-BFO [23]. In Table 1, *n* denotes the number of dimensions. The first benchmark function is unimodal in the sense that has only one optimum whereas the others are multimodal in the sense that many local optima with only one global optimum. All benchmark functions excluding Shekel's Foxholes function (f_6) have the global minimum at the origin whereas for f_6, the global minimum is at (-31.95,-31.95) and $f_6(-31.95,-31.95) \approx 0.998$. The information of test functions is summarized in table 2. The performance of aforementioned algorithms is evaluated along two dimensions: 1) the quality of optimum solution. Due to this reason, the mean and the standard deviation of the best-of-run values are considered, 2) the number of runs to find the optimum solution. In order to this, a threshold value must be defined as a stopping criterion.

Table 1. Test functions used

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

n represents the number of dimensions.

Table 2. Description of the test functions

Function	Trait	Search space	Initial rang	Global optimum
f_1	Unimodal	$[-30, 30]^n$	$[15, 30]^n$	$f_1(\vec{1}) = 0$
f_2	Multimodal	$[-5.12, 5.12]^n$	$[2.56, 5.12]^n$	$f_2(\vec{0}) = 0$
f_3	Multimodal	$[-600, 600]^n$	$[300, 600]^n$	$f_3(\vec{0}) = 0$
f_4	Multimodal	$[-30, 30]^n$	$[15, 30]^n$	$f_4(\vec{0}) = 0$
f_5	Multimodal	$[-65.536, 65.536]^2$	$[0, 65.536]^2$	$f_5(-31.95, -31.95) = 0.998$

To perform fair comparison, the same computational effort is used in all of the algorithms. Thereby, the same population size is considered as 40, for all algorithms. In BPSO, we set $c_1 = c_2 = 2$ and V_{max} and V_{min} are equal to the length of the search space. The inertia weight is also given 0.8. In BFO, we take $S = 40$, $N_s = 4$, $N_c = 50$, $N_{ed} = 1$, $N_{rd} = 4$, $P_{ed} = 0.25$, $d_{attract} = 0.1$, $\omega_{attract} = 0.2$, $\omega_{repellant} = 10$ and $h_{repellant} = 0.1$. The crossover probability P_c and the mutation probability P_m in GA are given 0.8 and 0.1, respectively. To evaluate the efficiency of the proposed APSO-BFO algorithm, it is compared on different dimensions sizes (n) of 15, 30, 45 and 60 with 25 independent runs. The number of iterations opted as a measure of computational time whereas different maximum numbers of fitness function evaluations were utilized according to the complexity of problem.

Simulation results are given in Tables 3-5. Table 3 represents the quality of the optimum solution of the algorithms by the mean and the standard deviation of best-of-run values whereas every algorithm is run up to a predetermined maximum number of fitness function evaluations. Comparative results indicated that the proposed PSO has good global search ability. During almost every run, the proposed algorithm can find the optimum of the complex test functions.

Table 4 represents the number of runs that managed to find the optimum solution within a prescribed threshold. For all benchmarks except Shekel's Foxholes function

(f_6), a threshold of 10^{-4} is considered as stopping criterion while for f_6, it was fixed at 0.998. Also, the maximum number of fitness function evaluations is set to 5000 if the algorithm can not reach to the threshold. Simulation results indicated the superiority of the proposed algorithm over the aforementioned algorithms. The results indicate in how many fitness function evaluations, the convergence of the solution or success is met. It is obvious that, the proposed algorithm spends extremely more fitness function evaluations to reach a predefined threshold as compared with other algorithms. Therefore, it can be concluded that APSO-BFO algorithm is superior to aforementioned algorithms in terms of accuracy.

5. Application of APSO-BFO for Optimal Control

The proposed APSO-BFO algorithm has a much more accuracy and faster convergence speed than aforementioned algorithms. Because of this, the proposed algorithm is applied to obtain optimal parameters of PID controller of nonlinear system.

5.1 Overview of PID Controller

The PID controller is the standard tool for industrial automation. The flexibility of the controller makes it possible to use PID control in many applications. The discrete control law of PID controller is

$$u(k) = u(k-1) + K_p[e(k) - e(k-1)] + K_i \frac{T_s}{2}[e(k) + e(k-1)] +$$
$$K_d \frac{1}{T_s}[e(k) - 2e(k-1) + e(k-2)] \tag{6}$$

where K_p, K_i and K_d are proportional gain, integral gain and derivative gain, respectively, e is the error signal between the desired output y_r and actual output y, u is the PID control force, and T_s is the sampling period.

88

Table 3. Convergence data for the benchmark functions

Name	Dim.	Max. Fitness Evaluations	Mean best value (Standard deviation)					
			SBFO	GA-BFO	DE-BFO	BPSO	PSO-BFO	APSO-BFO
f_1	15	5×10^4	26.6904 (2.163)	2.5029 (0.5409)	3.2103×10^{-3} (4.028×10^{-2})	14.222 (3.570)	0.4831 (0.0738)	2.2103×10^{-3} (0.328×10^{-2})
	30	10^5	58.215 (14.33)	22.4033 (1.4890)	0.26722 (5.746×10^{-2})	46.141 (9.646)	15.4708 (2.6530)	0.2251 (6.493×10^{-3})
	45	5×10^5	96.872 (26.135)	5.0508 (1.2996)	0.0449 (0.332×10^{-2})	83.629 (14.533)	27.9890 (4.3379)	0.0238 (0.112×10^{-3})
	60	10^6	154.702 (40.160)	25.4421 (0.2275)	0.91221 (0.04346)	122.241 (67.721)	52.2628 (8.3391)	0.8991 (0.0298)
f_2	15	5×10^4	6.9284 (2.091)	6.7382×10^{-3} (0.0001)	6.7382×10^{-3} (0.0001)	3.3479 (0.292)	0.0830 (0.00929)	6.7382×10^{-3} (0.0000)
	30	10^5	17.0389 (4.822)	0.04582 4.579×10^{-4}	6.7382×10^{-3} (0.0001)	12.7369 (0.779)	10.2257 (0.2405)	6.7382×10^{-3} (0.0001)
	45	5×10^5	30.9923 (7.8272)	0.1730 7.1050×10^{-4}	6.503×10^{-3} (4.221×10^{-2})	24.8279 (1.8190)	13.5031 (3.9235)	9.9763×10^{-5} (4.055×10^{-2})
	60	10^6	45.8235 (9.6222)	0.0833 (0.1152)	8.3627×10^{-4} (2.635×10^{-3})	36.3341 (6.2909)	18.3619 (5.7726)	5.7001×10^{-4} (2.225×10^{-3})
f_3	15	5×10^4	0.2811 (0.0212)	6.7382×10^{-3} (0.0000)	6.7382×10^{-3} (0.0000)	0.0363 (0.00531)	0.0531 (0.0277)	6.7382×10^{-3} (0.0000)
	30	10^5	0.3731 (0.0446)	0.0391 (0.0423)	6.7382×10^{-3} (0.0000)	0.1344 (0.1071)	0.0703 (0.0111)	6.7382×10^{-3} (0.0000)
	45	5×10^5	0.6352 (0.0519)	0.1715 (0.0712)	3.0071×10^{-4} (1.832×10^{-5})	0.1966 (0.1161)	0.1349 (0.0138)	2.111×10^{-4} (9.728×10^{-6})
	60	10^6	0.8322 (0.073)	0.5021 (0.5430)	3.2876×10^{-3} (1.536×10^{-3})	0.7588 (0.3431)	0.2539 (0.0283)	2.801×10^{-3} (1.009×10^{-3})
f_4	15	5×10^4	0.9333 (0.0285)	0.0751 (0.3083)	6.7381×10^{-3} (0.0000)	0.5819 (0.0540)	0.08249 (0.0007)	6.7382×10^{-3} (0.0000)
	30	10^5	4.3244 (1.8870)	0.3615 (0.0921)	6.7382×10^{-3} (0.0000)	0.8577 (0.0409)	0.5919 (0.0357)	6.7382×10^{-3} (0.0000)
	45	5×10^5	12.4562 (3.4333)	1.3179 (0.1303)	1.8171×10^{-4} (1.958×10^{-7})	1.8982 (0.1949)	0.9379 (0.1329)	1.8091×10^{-5} (1.880×10^{-7})
	60	10^6	8.3245 (1.6141)	1.1840 (0.7630)	2.2627×10^{-4} (6.347×10^{-5})	2.4061 (0.4508)	1.8769 (0.538)	2.0987×10^{-4} (9.347×10^{-6})
f_5	2	5×10^4	0.999866 (0.00215)	3.6791 (1430×10^{-6})	0.99980 (0.0000)	0.999829 (0.00166)	0.999800 (0.0000)	0.99980 (0.0000)

89

Table 4. Number of runs converging to the prescribed threshold

Function	Dim.	Number of runs					
		SBFO	GA-BFO	DE-BFO	BPSO	PSO-BFO	APSO-BFO
f_1	15	5	10	17	10	15	9
	30	0	0	12	0	12	13
	45	0	0	6	0	5	16
	60	0	0	3	0	4	10
f_2	15	0	25	25	25	25	25
	30	0	10	25	30	24	25
	45	12	17	25	14	25	25
	60	6	2	25	0	25	25
f_3	15	25	25	25	30	25	25
	30	10	12	25	15	21	25
	45	9	14	15	6	12	15
	60	4	9	17	8	16	17
f_4	15	16	15	25	13	30	25
	30	18	12	25	12	20	25
	45	9	9	19	5	20	22
	60	6	3	11	2	11	15
f_5	2	15	16	25	25	25	25

How to solve these three gains to meet the required performance is the most key in the PID control system. For simplification, let $\theta = [\theta_1, \theta_2, ..., \theta_n]^T = [K_p, K_i, K_d]^T$ be control gain vector. In the control process, three PID control gains (K_p, K_i, K_d) are calculated by the proposed algorithm such that the defined fitness function, defined as the Sum of Square of Errors (SSE), is minimized. In the PID controller design, the fitness function is modified by

$$SSE = \sum_{k=1}^{N} e^2(k) = \sum_{k=1}^{N}(y_r(k) - y(k))^2 \qquad (7)$$

5.2 Design of PID Controller

To evaluate the feasibility of the proposed algorithm, it is employed to determine optimal tuning of the PID controller parameters for control of nonlinear systems. In this example, we verify of the design approach on the tracking control of a Duffing forced oscillation system. It is noticeable that the system is chaotic without control. The dynamic equation of the system is as follows [24]:

$$\dot{x}_1 = x_2$$
$$\dot{x}_2 = -0.1x_2 - x_1^3 + 12\cos(t) + u \qquad (8)$$

In Eq. (8), *u* denotes the control input calculated by PID controller. The parameters were chosen based on values represented in Section 4. It is noticeable that in this case, the dimension of search space *P* is 3. In this simulation, the control objective is to wish that the plant output *y* is regulated to the desired output $y_r = \sin(t)$. The search space for PID gains is defined by

$$K_{p\min} = 0 , \ K_{p\max} = 100 , \ K_{d\min} = 0 , \ K_{d\max} = 10, \ K_{i\min} = 0 , \ K_{i\max} = 10$$

Table 5 shows the results obtained for PID Controller parameters for BFO, BPSO, GA-BFO, DE-BFO, PSO-BFO and APSO-BFO algorithms. Results depict that the

90

proposed algorithm has better performance than others. Moreover, Table 6 represents the performance comparison the aforementioned algorithm by worst, mean, best and standard deviation of SSE results during 30 runs for each algorithm. Results show that the proposed algorithm has satisfactory performance. The output response of system is depicted in Fig. 1. It is clearly obvious that the proposed algorithm has superior features, including fast tuning and accuracy. Fast tuning of optimal PID controller parameters yields high-quality solutions.

Table 5. Obtained parameters of PID controller for Duffing system

PID Parameters	K_p	K_i	K_d
BFO	50.8411	1.9289	2.0057
GA-BFO	49.8282	1.0305	2.101
DE-BFO	52.5436	1.4044	1.0426
BPSO	52.5780	1.4728	1.4350
PSO-BFO	51.8543	1.4845	1.3535
APSO-BFO	50.6733	1.4838	1.3510

Table 6. Performance comparison of PID controller for Duffing system

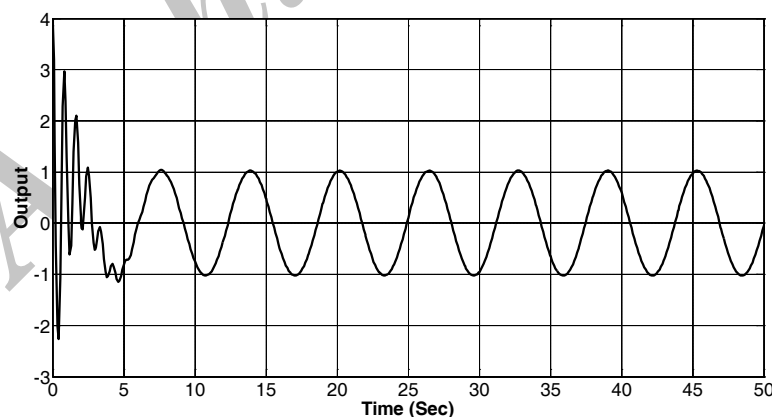| Criterion	Worst	Mean	Best	Standard deviation
BFO	1.5967	1.2008	1.9194	0.1688
GA-BFO	1.0468	1.0169	1.0097	0.0994
DE-BFO	1.0209	0.9887	0.5051	0.0583
BPSO	1.5616	1.1304	0.9540	0.1305
PSO-BFO	0.1989	0.1090	0.0081	0.0456
APSO-BFO	0.1545	0.0914	0.0018	0.0088

Figure 1. The output response of Duffing system after applying PID controller by APSO-BFO algorithm

6. Conclusion

This paper proposed a novel heuristic algorithm by integrating APSO and BFO, namely APSO-BFO algorithm. The main advantages of the proposed algorithm are to overcome the inherent problems of both the BPSO and BFO. The proposed hybrid

91

algorithm performs local search through the chemotactic movement operation of BFO whereas the APSO accomplishes the global search over the entire search space. The superiority of the proposed algorithm both in robustness and efficiency was verified over a few numerical benchmarks and a practical PID tuner design problem. The results clearly reveal the effectiveness of the proposed algorithm in more robust and accurate than BPSO, SBFO, PSO-BFO, GA-BFO and DE-BFO.

7. References

[1] Astrom, K. J., Wittenmark, B.: Adaptive control. Addison-Wesley, Massachusetts (1995).

[2] Coelho, L.D.S., Bernert, D.L.A.: PID control design for chaotic synchronization using a tribes optimization approach. Chaos Solit. Fract. 42(1), 634--640 (2009).

[3] Huang, H.P., Roan, M.L., Jeng, J.C.: On-line adaptive tuning for PID controllers. Control Theory Appl. 149(1), 60--67 (2002).

[4] Cominos, P., Munro, N.: PID controllers: recent tuning methods and design to specification. In: IEE Control Theory and Applications, pp: 46--53 (2002).

[5] Astrom, K.J., Hagglund, T.H.: PID controllers: theory, design and tuning, International Society for Measurement and Control (1995).

[6] Alfi, A., Fateh, M.M.: Intelligent identification and control using improved fuzzy particle swarm optimization. Expert Syst. Appl. 38, 12312--12317 (2011).

[7] Modares, H., Alfi, A., Fateh, M.M.: Parameter identification of chaotic dynamic systems through an improved particle swarm optimization. Expert Syst. Appl. 37, 3714--3720 (2010a).

[8] Valarmathi, K., Devaraj, D., Radhakrishnan, T.K.: Real-coded genetic algorithm for system identification and controller tuning. Appl. Math. Model. DOI:10.1016/j.apm.2008.11.006 (2008).

[9] Chang, W.D.: Nonlinear system identification and control using a real-coded genetic algorithm. Appl. Math. Model. 31, 541--550 (2007).

[10] Eberhart, R.C., Shi, Y.: Comparison between genetic algorithms and particle swarm optimization. Lecture Notes in Computer Science, pp: 611--618 (1998).

[11] Bae, C., Yeh, W.C., Chung, Y.Y., Liu, S.L.: Feature selection with intelligent dynamic swarm and rough set. Expert Syst. Appl. DOI: 10.1016/j.eswa.2010.03.016 (2010).

[12] Yeh, W.C.: A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems. Expert Syst. Appl. 36, 9192--9200 (2009).

[13] Lin, Y.L., Chang, W.D., Hsieh, J.G.: A particle swarm optimization approach to nonlinear rational filter modeling. Expert Syst. Appl. 34, 1194--1199 (2008).

[14] Mansouri, R., Bettayeb, M., Djamah, T., Djennoune, S.: Vector fitting fractional system identification using particle swarm optimization. Appl. Math. Model. 206(2), 510--520 (2008).

[15] Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm intelligence, Morgan Kaufmann Publishers, San Francisco (2001).

[16] Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Systems Magazine, 52--67 (2002).

[17] Mishra, S.: A hybrid least square-fuzzy bacterial foraging strategy for harmonic estimation. IEEE Trans. Evol. Comput. 9(1), 61--73 (2005).

[18] Kim, D.H., Cho, C.H.: Bacterial foraging based neural network fuzzy learning. In: Indian International Conference on Artificial Intelligence, pp: 2030--2036 (2005).

[19] Tripathy, M., Mishra, S., Lai, L.L., Zhang, Q.P.: Transmission loss reduction based on FACTS and bacteria foraging algorithm. In: 9th International Conference on Parallel Problem Solving from Nature, pp: 222--231 (2006).

[20] Modares, H., Alfi, A., Naghibi Sistani, M.B.: Parameter estimation of bilinear systems based on an adaptive particle swarm optimization. J. Eng. Appl. Artif. Intel. 23, 1105--1111 (2010b).

[21] Kim, D.H., Abraham, A., Cho, J.H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization. Info. Sci. 177(18), 3918--3937 (2007).

[22] Biswas, A., Dasgupta, S., Das, S., Abraham, A.: Synergy of PSO and bacterial foraging optimization: A comparative study on numerical benchmarks. In: Second International Symposium on Hybrid Artificial Intelligent Systems, pp: 255--263 (2007a).

[23] Biswas, A., Dasgupta, S., Das, S., Abraham, A.: A synergy of differential evolution and bacterial foraging algorithm for global optimization. Int. J. NeuralMass-Parallel Comput. Inf. Syst. 17(6), 607--626 (2007b).

[24] Wang, L.X.: Adaptive fuzzy systems and control: design and stability analysis, Prentice Hall, Englewood Cliffs, NJ (1995).