World Applied Sciences Journal 22 (12): 1782-1788, 2013
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2013.22.12.2606

Design of Fuzzy Controller for PUMA 560 Robot Arm Using Improved Bacterial Foraging Optimization Algorithm

Mickael Aghajarian and Kourosh Kiani

Department of Electrical and Computer Engineering, Semnan University, Semnan, Iran

Abstract: Trial and error method can be used to find a suitable design of a fuzzy controller. Generally, the design of fuzzy controller involves determination of the fuzzy rules, Membership Functions (MFs) and scaling factors. An optimization algorithm facilitates the design process and finds an optimal design to achieve a desired performance. This paper presents an Improved Bacterial Foraging Optimization Algorithm (IBFOA) to design a fuzzy controller for tracking control of a PUMA 560 robot arm driven by permanent magnet DC motors. We use efficiently the IBFOA to form the rule base and MFs. To show the improvement of proposed algorithm, the IBFOA is compared with Bacterial Foraging Optimization Algorithm (BFOA) and Particle Swarm Optimization (PSO) algorithm. Performance of the controller in the joint space and in the Cartesian space is evaluated. Simulation results show superiority of the IBFOA to the BFOA and PSO algorithm.

Key words: Evolutionary Algorithms · Fuzzy Logic · PUMA 560 Robot Arm · Tracking Control

INTRODUCTION

A wide variety of control strategies were proposed to control robot manipulators. PID controls are certainly the most widely adopted control strategy in industry because of its simple structure and robust performance in a wide range of operating conditions. Although PID control offers the simplest and yet most efficient solution to many real world control problems [1], optimally tuning gain is quite difficult [2]. Alternatively, fuzzy control as a model-free approach is simply designed to control complicated systems [3]. To form fuzzy rules, an exact knowledge of model is not required. Fuzzy controller is an intelligent controller using linguistic fuzzy rules to include information from experts. Consequently, fuzzy control of robot manipulators has attracted a great deal of researches to overcome uncertainty, nonlinearity and coupling by providing a model-free control [4]. To design a Fuzzy Logic Controller (FLC), a major task is to determine fuzzy rules, Membership Functions (MFs) and scaling factors. Therefore, the controller is tuned until a desired performance is achieved. An optimization algorithm facilitates the design process and finds an optimal design to achieve a desired performance. In the last decade, approaches based on evolutionary algorithms have received increased attention from the researchers for solving optimization problems that could not be solved using conventional problem solving techniques [5]. Recently natural swarm inspired algorithms like Bacterial Foraging Optimization Algorithm (BFOA) [6], Particle Swarm Optimization (PSO) [7] and Ant Colony Optimization (ACO) [8] have been dominating the realm of optimization algorithms and proved their effectiveness.

Since a foraging organism or animal takes a necessary action to maximize the energy obtained per unit time spent for foraging, in the face of constraints presented by its own physiology, such as sensing and cognitive capabilities, natural foraging strategy can be applied to real-world optimization problems. Based on such evolutionary idea, Passino proposed BFOA as an optimization algorithm [6]. BFOA is a new evolutionary computation technique, which also includes powerful optimization techniques like PSO [7] and ACO [8]. To improve BFOA search performance, several researchers have extended the basic BFOA to deal with multi-modal and high dimensional functions [9-11]. Researchers are trying to hybridize BFOA with other evolutionary algorithms [12-13] in order to reduce the convergence time and enhance the accuracy. Over certain real-world optimization problems, BFOA has been reported to outperform many powerful optimization algorithms like genetic algorithm [14] and PSO algorithms [15].

Corresponding Author: Mickael Aghajarian, Department of Electrical & Computer Engineering, Semnan University, Semnan, Iran. Tel: +98 (913) 3051982.

The PSO algorithm proposed by Kennedy and Eberhart [7], has proved to be very effective for solving complex optimization problems. The underlying motivation for the development of PSO algorithm was social behavior of animals such as bird flocking and fish schooling. Generally, PSO is characterized as a simple concept, easy to implement and computationally efficient. Unlike the other heuristic techniques, PSO has a flexible and well-balanced mechanism to enhance the global and local exploration abilities [16].

Trial and error method is a major task to find a suitable design of a fuzzy controller. We may use an optimization algorithm to achieve an optimal design. This paper presents an IBFOA to design a fuzzy PID controller for trajectory tracking control of a PUMA 560 robot arm driven by permanent magnet DC motors. Performance of the IBFOA is compared with BFOA and PSO algorithm in terms of Integral Time Absolute Error (ITAE) in the joint space as well as Cartesian space. This paper is organized as follows: Section 2 introduces dynamics of the robotic system. Section 3 designs a fuzzy PD+I controller. Section 4 applies the IBFOA, BFOA and PSO algorithms for tuning the fuzzy PD+I controller. Section 5 presents simulation results and Finally Section 6 concludes the paper.

Manipulator Dynamics: Robust control of robot manipulators is difficult because of complexity robot dynamics. The dynamics of an n-link robotic manipulator driven by permanent magnet DC motors is characterized by a set of highly nonlinear and strongly coupled second order differential equation [17] as

$$RK_m^{-1}\left(J_m r^{-1}+rD(q)\right)\ddot{q}+\left(RK_m^{-1}B_m r^{-1}+RK_m^{-1}rC(q,\dot{q})+K_b r^{-1}\right)\dot{q} + RK_m^{-1}rg(q)=V \quad (1)$$

where $q \in R^n$ is a vector of generalized joint positions, $D(q) \in R^{n\times n}$ is the inertia matrix, $C(q,\dot{q})\dot{q} \in R^n$ is a vector of centripetal and Coriolis generalized forces, $g(q) \in R^n$ is a vector of generalized gravitational forces, $V \in R^n$ is a vector of motor voltages, K_m, K_b, R, J_m, B_m, $r \in R^{n\times n}$ are constant diagonal matrices of torque constant, back emf constant, resistance, inertia, damping and reduction gear ratio of motors, respectively.

Fuzzy Pd+i Controller: Fuzzy PID controller is implemented as fuzzy PD+I controller as shown in Figure 1. Each fuzzy set consists of a number of MFs to describe the heuristic variables in a mathematical manner. The motor voltage is the output of fuzzy controller while the joint position error and its derivative are its inputs. MFs for the inputs and output of the controller are five fuzzy sets namely NB (Negative Big), NM (Negative Medium), Z (Zero), PM (Positive Medium) and PB (Positive Big). The following assumptions are given to design the FLC:

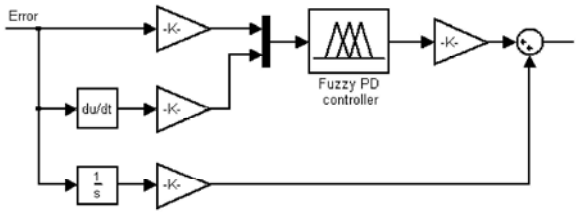

Fig. 1: Fuzzy PD+I controller.

- Mfs are triangular specified with three points, S-shaped built-in MF and Z-shaped built-in MF that specified with two points.
- The physical range of inputs are scaled between [–1 1].
- Number of fuzzy sets is an odd integer greater than one.
- The first and the last MF are Z-shaped built-in MF and S-shaped built-in MF, respectively.
- The extremes of the sloped portion of the Z-shaped built-in MF are at -1 and the second point of the adjacent triangular MF.
- The extremes of the sloped portion of the S-shaped built-in MF are at 1 and the second point of the adjacent triangular MF.
- Triangular MFs are arranged such that the second point of each triangular MF is coincident with the third point of the left one and the first point of the right one.

Then, number and position of second point of triangular MFs are selected as two design parameters. Position of the MF is specified by spacing parameter where one indicates an even spacing, while any value larger than unity indicates that the MFs are close together in the center of the range and more spaced out at the extremes as shown in Figure 2.

The rule-base is designed based on the ideas presented in [18]. In specifying a rule base, characteristic spacing parameter for each variable and characteristic angle for each input variable are used to construct the rules. The characteristic spacing parameters and the characteristic angle determine how the space is partitioned. The angle determines the slope of a line

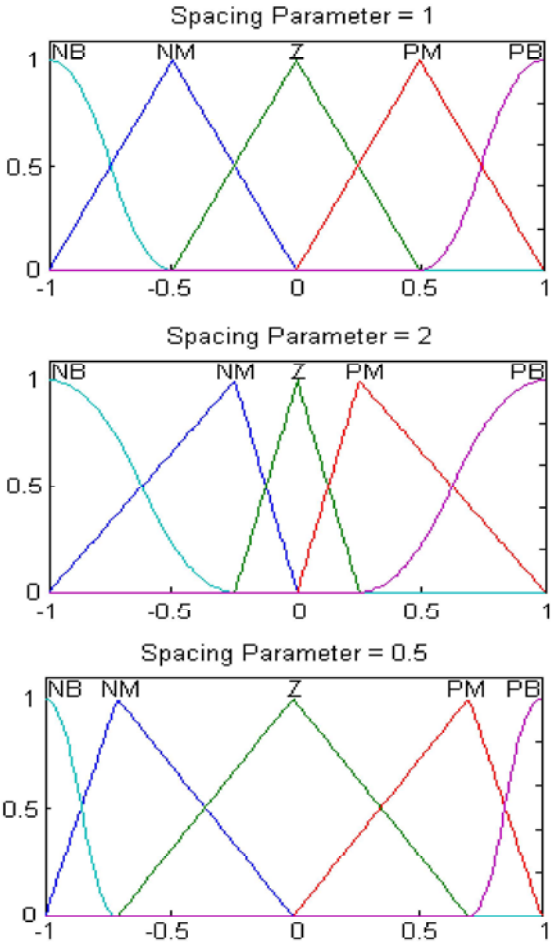

Fig. 2: Effect of spacing parameter on MFs.

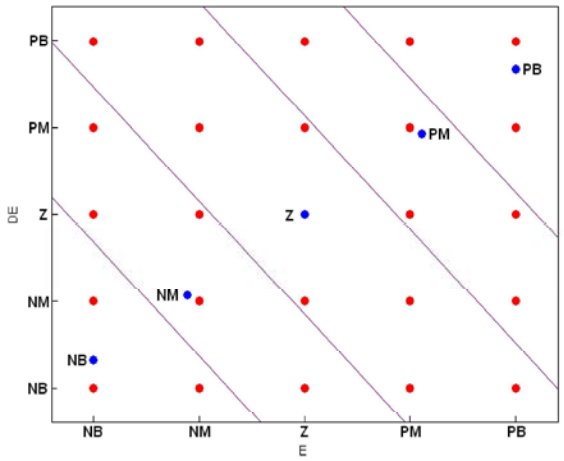

Fig. 3: Sample decision plane.

through the origin on which seed points are placed, where seed points are blue circles and grid-points are red circles in Figure 3. The positioning of the seed points is determined by a similar spacing method as was used to determine the centers of the Mfs as illustrated in Figure 3. The lines on the graph delineate the different regions corresponding to different consequents. The parameters for this example are 1 for both input spacing, 0.85 for the output spacing and 40° for the angle. Table 1 shows the derived fuzzy rules.

Table 1: Fuzzy rules

		E				
		NB	NM	Z	PM	PB
DE	NB	NB	NB	NM	Z	Z
	NM	NB	NM	Z	Z	PM
	Z	NM	NM	Z	PM	PM
	PM	NM	Z	Z	PM	PB
	PB	Z	Z	PM	PB	PB

Bacterial Foraging Optimization Algorithm: The BFOA can be explained by four processes namely, chemotaxis, swarming, reproduction and elimination-dispersal [6]. Below we briefly describe each of these processes.

Chemotaxis: This process simulates the swimming and tumbling movements of an E. coli cell by a set of rigid flagella. An E. coli bacterium can move in two different ways. It can swim for a period in the same direction or it may tumble and alternate between these two modes of operation for the entire lifetime. This alternation between the two modes enables the bacterium to move in random directions and search for nutrients. Suppose $\theta^i(j,k,l)$ represents *i-th* bacterium at *j-th* chemotactic, *k-th* reproductive and *l-th* elimination-dispersal step. *C(i)* is the run length which is a constant in basic BFOA. In computational chemotaxis, the movement of the bacterium is represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\Delta(i)/\sqrt{\Delta^T(i)\Delta(i)} \quad (2)$$

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

If the cost function value at $\theta^i(j+1, k,l)$ is better than the cost function value at $\theta^i(j,k,l)$, then the bacterium takes another step of size $C(i)$ in this same direction and again, if that step resulted in a position with a better cost function value than at the previous step, another step is taken. This swim is continued until the number of steps taken is greater than the maximum number of steps, *Ns*.

Swarming: It is always desired that when any one of the bacteria reaches the better location, try to attract other bacteria so that they reach the desired place more rapidly.

The effect of swarming is to make the bacteria congregate into groups and move as concentric patterns with high bacterial density. Mathematically, swarming can be represented by

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l)) = \sum_{i=1}^{S}\left[-d_{attrac\tan t}\exp\left(-w_{attrac\tan t}\sum_{m=1}^{p}(\theta_m-\theta_m^i)^2\right)\right] + \sum_{i=1}^{S}\left[-h_{repellant}\exp\left(-w_{repellant}\sum_{m=1}^{p}(\theta_m-\theta_m^i)^2\right)\right] \quad (3)$$

where $J_{cc}(h, P(j, k, l))$ represents the objective function value to be added to the actual objective function (to be minimized) to present a time varying objective function, S is the total number of bacteria, p is the number of variables to be optimized, which are present in each bacterium and $\theta = [\theta_1, \theta_2, ..., \theta_p]^T$ is a point in the p-dimensional search domain. $d_{\text{attractant}}$, $w_{\text{attractant}}$, $h_{\text{repellant}}$, $w_{\text{repellant}}$ are different coefficients that should be chosen properly.

Reproduction: The least healthy bacteria eventually die while each of the healthier bacteria (each with the lower cost function) asexually split into two bacteria, which are then placed in the same location. Thus, the population size after reproduction is maintained constant.

Elimination and Dispersal: A gradual or sudden changes in the location where a bacterium population lives may occur due to noxious substance, the temperature rises abruptly in the area or some other influence. Events can kill or disperse all the bacteria in a region. This reduces the chances of convergence at local optima location. To simulate this phenomenon in BFOA, some bacteria are chosen, according to a preset probability P_{ed}, to be dispersed and moved to another position within the environment.

The BFOA parameters are denoted as p, S, N_c, N_s, N_{re}, N_{ed}, P_{ed}, where p is dimension of the search space, S is the total number of bacteria in the population, N_c is number of chemotactic steps, N_s is swimming length, N_{re} is number of reproduction steps, N_{ed} is number of elimination-dispersal events, P_{ed} is elimination-dispersal probability.

Improved Bacterial Foraging Optimization Algorithm: The flaw in BFOA is, constant step size *C(i)*, which affects the speed of convergence and precision. If the step size is very high, there is a chance for the bacterium to miss the accurate values, although they reach the vicinity of optimum point quickly. If the step size is very small, then it takes many chemotactic steps to reach the optimum position. Therefore, the speed of convergence decreases. Hence, the step size of each bacterium is the main determining factor for both the speed of convergence and precision. A step size varying as the function of the current cost function, is expected to provide better convergence and precision as compared to a fixed step size. If deviation is too large then the step size must be increased and if the deviation is small in which case the bacterium is close to the optimum position the step size is to be reduced.

In this paper, we have used the fuzzy system to control the step size. The cost function value and step size are considered as the input and output of the fuzzy system, respectively. MFs for the inputs and output of the fuzzy system are five fuzzy sets namely Z (Zero), PVS (Positive Very Small), PS (Positive Small), PM (Positive Medium) and PB (Positive Big). The fuzzy system is then simply designed as follows

If *J(i,j,k,l)* is Z, then *C(i)* is Z.
If *J(i,j,k,l)* is PVS, then *C(i)* is PVS.
If *J(i,j,k,l)* is PS, then *C(i)* is PS.
If *J(i,j,k,l)* is PM, then *C(i)* is PM.
If *J(i,j,k,l)* is PB, then *C(i)* is PB.

where *J(i,j,k,l)* is the cost function value for the *i*-th bacterium at *j-th* chemotactic, *k-th* reproductive and *l-th* elimination-dispersal step.

According to BFOA, the bacterium swims only if the new cost function value is less than the previous one, *i.e. J(i,j+1,k,l) < J(i,j,k,l)*. This swim is continued until the number of steps taken is greater than the maximum number of steps, *Ns*.

In our proposed algorithm, if *J(i,j+1,k,l) > J(i,j,k,l)*, the bacterium swims according to the probability distributions that specified as Boltzmann annealing. The acceptance probability defines as follows

$$\text{Pr}ob = \exp(-\Delta J/T) \quad (4)$$

where ΔJ represents the difference between the present and previous values of the cost function, *i.e.*,

ΔJ = *J(i,j+1,k,l)* - *J(i,j,k,l)* and T is a control parameter called temperature. Clearly, at sufficiently high values of the control parameter T, the acceptance probability is high. As T decreases, the acceptance probability becomes smaller. The cooling rate is defined

by $Ti+1 = k * Ti$, where k is a positive constant smaller than 1. The parameters selected for the proposed IBFOA are shown in Table 2.

Particle Swarm Optimization: PSO is a population-based optimization method inspired by the social behavior of animals such as bird flocking and fish schooling. Like evolutionary algorithms, PSO algorithm conducts search using a population of particles, corresponding to individuals. Each particle has a velocity vector v^i and a position vector x^i to represent a possible solution to the optimization problem. The first positions and velocities of a PSO algorithm are randomly initialized within a population. At the next iteration, position and velocity of each particle are updated by the two values. The first value, pbest, is the personal best position of particle that it has achieved so far. The other, gbest, is obtained by choosing the overall best value from all particles. The new velocity for each particle is updated by the following equation

$$v_n^{(t+1)} = wv_n^{(t)} + c_1 r_1 \left(\text{pbest}_d - x_d^{(t)} \right) + c_2 r_2 \left(\text{gbest}_d - x_d^{(t)} \right) \quad (5)$$

where w, c_1 and c_2 are called the coefficient of inertia, cognitive and society study, respectively. The r_1 and r_2 is uniformly distributed random numbers in [0, 1]. Changing velocity enables every particle to search around its individual best position and global best position. Based on the updated velocities, each particle changes its position as following

$$x_n^{(t+1)} = x_n^{(t)} + \chi v_n^{(t)} \quad (6)$$

Optimization of FLC Using IBFOA: In this paper, spacing parameter for MFs of input/output variables, spacing parameters and angle parameters for rule base and input/output scaling factors of FLCs are determined using IBFOA. Figure 4. illustrates the block structure of the FLC optimizing process using IBFOA. All parameters of the FLC are updated at every final time (t_f). The method of tuning PID parameters is based upon minimizing the *ITAE* of joints. If $q_d(k)$ is desired trajectory and $q(k)$ is output trajectory then error $e(k)$ is

$$e(k) = q_d(k) - q(k) \quad (7)$$

$$ITAE = \sum_{j=1}^{3} \sum_{k=1}^{n} \left| e(kj).kj \right| \quad (8)$$

Table 2: Parameters of IBFOA

S	N_c	N_s	N_{re}	N_{ed}	P_{ed}	k	T_1
10	30	4	3	3	0.25	0.85	10000

Table 3: Parameters of permanent magnet DC motors

Joint	R	L	K_b	J	B	1/r
1	1.6	0.005	0.26	0.0002	0.00148	62.611
2	1.6	0.005	0.26	0.0002	0.000817	107.82
3	1.6	0.005	0.26	0.0002	0.00138	53.706

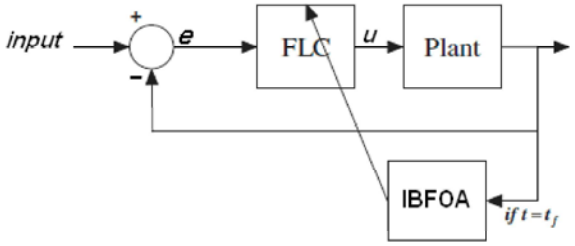

Fig. 4: Tuning of FLC parameters using IBFOA.

where $e(kj)$ is the system error at k-th sampling instant for j-th joint. Tuning process of FLC parameters using BFOA and PSO is similar to IBFOA.

Simulation Results: The control system is simulated for tracking control of PUMA 560 robot arm driven by brushed permanent magnet DC motors with specifications [19] given in Table 3. Motors, which drive joints 1, 2 and 3 are 40 V and 160 W [20]. The simulation model of PUMA 560 in MATLAB [21] is used in the control system.

Results of the tuning methods are tested in term of ITAE in joint space as well as Cartesian space. The ability of control system for rejecting disturbances is simulated. For checking the robustness of controller a disturbance torque D is applied as an example in the form of

$$D = 3\sin(4t) + 4 \quad (9)$$

Fitness function curve of PSO tuning is shown in Figure 5.

Performance of the second joint for tracking desired trajectory and corresponding error are shown in Figure 6. and Figure 7., respectively. Initial angles of all three joints are set to zero, however desired trajectory is starting from a non-zero value in Figure 6.

We can see a good tracking, where the tracking error of the joint is very small. Table 4 and Table 5 give the values of the ITAE cost function with and without disturbance for the controllers. As can be seen, the Fuzzy-IBFOA based controller is better than other two controllers. Considering the results confirm a powerful

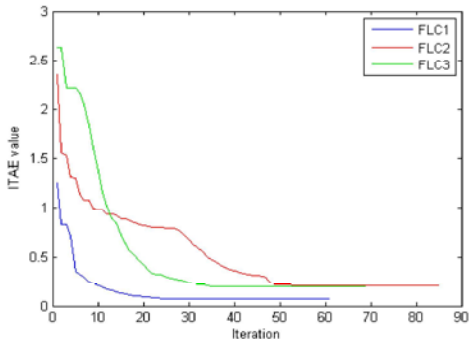

Fig. 5: Fitness function in PSO algorithm.

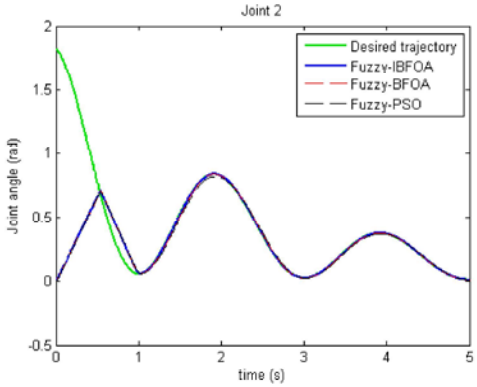

Fig. 6: Performance of second joint angle.

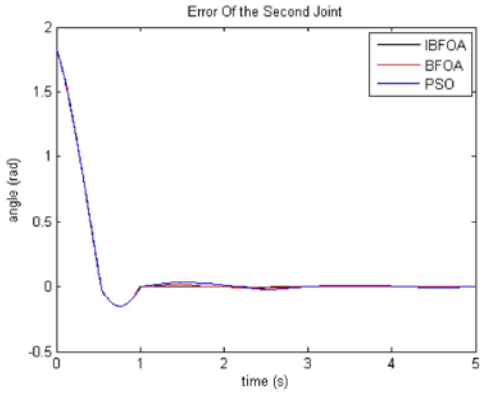

Fig. 7: Joint space error of second joint.

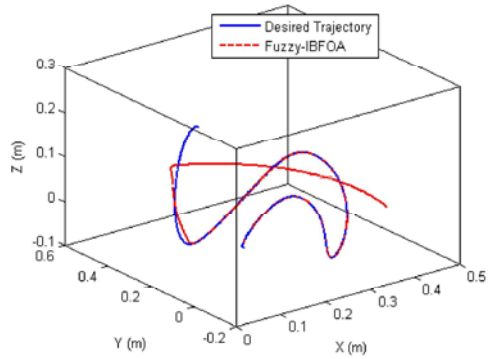

Fig. 8: Performance of FLC using IBFOA.

Table 4: Cost function (ITAE) in joint space

	Without Disturbance	With Disturbance
IBFOA	0.4210	0.4287
BFOA	0.4545	0.4620
PSO	0.5236	0.5311

Table 5: Cost function (ITAE) in Cartesian space

	Without Disturbance	With Disturbance
IBFOA	0.0721	0.0729
BFOA	0.0803	0.0812
PSO	0.1034	0.1041

ability of rejecting disturbances. Figure 8. shows desired and tracked trajectory for the Fuzzy-IBFOA based controller. With comparing performances in simulations, it can be concluded that the IBFOA is superior to the BFOA and PSO algorithm in term of accuracy of response.

CONCLUSION

In this paper, we proposed an IBFOA and compared it with BFOA and PSO algorithms to design a fuzzy controller for tracking control of a PUMA 560 robot arm. Performances of controllers in the cases of with and without disturbances are compared for the above approaches in joint space, as well as in Cartesian space. The simulation results show that IBFOA is superior to the BFOA and PSO algorithm in term of accuracy of response. An improvement of this work can be made by designing an online adaptive controller based on IBFOA.

REFERENCES

1. Ang, K.H., G. Chong and Y. Li, 2005. PID Control System Analysis, Design and Technology. IEEE Transactions on Control Systems Technology, 13(4): 559-576.
2. Kim, T.H., I. Maruta and T. Sugie, 2008. Robust PID Controller Tuning Based on the Constrained Particle Swarm Optimization. Automatica, 44(4): 1104-1110.
3. Wang, L.X., 1996. A Course in Fuzzy Systems and Control. Prentice Hall.
4. Fateh, M.M., 2010. Robust Fuzzy Control of Electrical Manipulators. Journal of Intelligent and Robotic Systems, 60(3-4): 415-434.
5. Fleming, P.J. and R.C. Purshouse, 2002. Evolutionary Algorithms in Control System Engineering: A Survey. Control Engineering Practice, 10(11): 1223-1241.

6. Passino, K.M., 2002. Biomimicry of Bacterial Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine, 22(3): 52-67.
7. Kennedy, J. and R. Eberhart, 1995. Particle Swarm Optimization. In the IEEE International Conference on Neural Networks, pp: 1942-1948.
8. Dorigo, M. and T. Stutzle, 2004. Ant Colony Optimization. MIT Press.
9. Tripathy, M. and S. Mishra, 2007. Bacteria Foraging-based to Optimize Both Real Power Loss and Voltage Stability Limit. IEEE Transactions on Power Systems, 22(1): 240-248.
10. Munoz, M.A., J.A. Lopez and E. Caicedo, 2007. Bacteria Swarm Foraging Optimization for Dynamical Resource Allocation in A Multizone Temperature Experimentation Platform. Analysis and Design of Intelligent Systems using Soft Computing Techniques, Advances in Intelligent and Soft Computing, 41: 427-435.
11. Shen, H., Y. Zhu, X. Zhou, H. Guo and C. Chang, 2009. Bacterial Foraging Optimization Algorithm with Particle Swarm Optimization Strategy for Global Numerical Optimization. In the Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp: 497-504.
12. Kim, D.H., A. Abraham and J.H. Cho, 2007. A Hybrid Genetic Algorithm and Bacterial Foraging Approach for Global Optimization. Information Sciences, 177(18): 3918-3937.
13. Biswas, A., S. Dasgupta, S. Das and A. Abraham, 2007. A Synergy of Differential Evolution and Bacterial Foraging Algorithm for Global Optimization. Neural Network World, 17(6): 607-626.
14. Su, T.J., G.Y. Chen, J.C. Cheng and C.J. Yu, 2010. Fuzzy PID Controller Design Using Synchronous Bacterial Foraging Optimization. In the 3rd International Conference on Information Sciences and Interaction Sciences, pp: 639-642.
15. Biswas, A., S. Dasgupta, S. Das and A. Abraham, 2007. Synergy of PSO and Bacterial Foraging Optimization: A Comparative Study on Numerical Benchmarks. Innovations in Hybrid Intelligent Systems, Advances in Intelligent and Soft Computing, 44: 255-263.
16. Clerc, M. and J. Kennedy, 2002. The Particle Swarm-Explosion, Stability and Convergence in A Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation, 6(1): 58-73.
17. Fateh, M.M., 2010. Proper Uncertainty Bound Parameter to Robust Control of Electrical Manipulators Using Nominal Model. Nonlinear Dynamics, 61(4): 655-666.
18. Park, Y.J., H.S. Cho and D.H. Cha, 1995. Genetic Algorithm-Based Optimization of Fuzzy Logic Controller Using Characteristic Parameters. In the IEEE International Conference on Evolutionary Computation, pp: 831-836.
19. Corke, P.I. and B. Armstrong-Helouvry, 1994. A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot. In the Proceedings of the IEEE International Conference on Robotics and Automation, pp: 1608-1613.
20. Wyeth, G.F., J. Kennedy and J. Lillywhite, 2000. Distributed Digital Control of A Robot Arm. In the Proceedings of the Australian Conference on Robotics and Automation, pp: 217-222.
21. Corke, P.I., 1996. A Robotics Toolbox for MATLAB. IEEE Robotics & Automation Magazine, 3(1): 24-32.