New Evolutionary Methods for Optimal Design of PID Controllers for AVR System

S. M. A. Mohammadi, A.A. Gharaveisi, M. Mashinchi, and S.M.R. Rafiei, Senior Member, IEEE

Abstract -An efficient and powerful design method suitable for calculating optimal proportional-integral-derivative (PID) controllers for AVR systems is proposed. The method is an improved version of the *Discrete Action Reinforcement Learning Automata* (DARLA) while discrete probability functions (DPF) of the design variables are not considered independent. The results of the proposed method called Extended *Discrete Action Reinforcement Learning Automata* (EDARLA) are compared to the results obtained by the well known Ziegler-Nichols (ZN), conventional DARLA and Genetic Algorithms (GA) approaches. The extensive simulation results prove superiority of the proposed design method in terms of optimality, efficiency, computation burden, and being less sensitive to the ranges considered for the design variables that is the search space. Besides being successful in providing globally optimal results, due to high efficiency and lower computation time, the proposed approach can be considered an interesting candidate for designing and tuning optimal adaptive PID controllers for many practical systems.

Key words*:* DARLA, CARLA, GA, PID Controller, Reinforced Learning, AVR System

I. INTRODUCTION

Reinforcement learning (RL) approaches are new but quite promising approaches giving a new scientific insight into the intelligent systems area with immense practical applications [1]-[16]. Reinforcement learning is different to supervised learning, which is a kind of learning being widely studied in current research in machine learning, statistical pattern recognition, and artificial neural networks. Supervised learning basically means learning from examples provided by a knowledgeable external supervisor. This is an important kind of learning, but it alone is not adequate without learning from interactions. Interactive problems are often off-line. An agent with its goal embedded in an environment learns how to transform one environmental state into another [6]. The agents with the ability of performing this task with minimal human supervision are called autonomous. Learning from an environment is more robust because agents are directly affected by the dynamics of the environment (the system under control).

Generally, adapting and tuning of the process parameters can be performed by either *Continuous Action Reinforcement Learning Automata* (CARLA)[17], or *Discrete Action Reinforcement Learning Automata* (DARLA) [7]. DARLA uses discrete action space making it more appropriate for the discrete engineering applications. CARLA was actually developed as an extension of the discrete stochastic learning automata methodology. Both DARLA and CARLA operate through interactions with a random or unknown environment by selecting actions in a stochastic trial and error process. CARLA replaces the discrete action space with a continuous one, using continuous probability distributions and hence making it more appropriate for engineering applications with continuous time variables. The only interconnection mechanism between DARLA is provided through the environment and via a shared performance or evaluation function. In each iteration, every action has an associated discrete probability functions (DPF) being used as a basis for its evaluation. The calculation is done separately for *'n'* disjoint DPFs where *'n'* is the number of parameters must be tuned so that an index function is optimized.

In this work we consider an 'n- *dimensional'* search space for the state of the environment (system) with a joint multi-variable discrete probability function for each cell in the search space which is updated at discrete samples. As will be shown, DARLA and CARLA methods significantly suffer from being too sensitive to the ranges considered for the design variables. If the ranges are too small, then there will be low chance to reach globally optimal results. On the other side, having large sets for the variables makes the method time consuming. The proposed approach (EDARLA) is potentially promising to achieve better results, and can more easily be adjusted so that the speed of convergence is significantly improved.

Numerous control methods such as fuzzy control, adaptive control, and neuro-fuzzy control have been studied [13]–[17]. Among them, the best known is the proportional-integral-derivative (PID) controller, which has been widely used in the industry because of its simple structure and robust performance in a wide range of operating

S.M.A Mohammadi and A.A. Gharaveisi are with the Department of Electrical Engineering, Shahid Bahonar University of Kerman, Kerman, Iran
M. Machinchi is with the Department of Mathematics, Shahid Bahonar University of Kerman, Kerman, Iran
S.M.R. Rafiei is with the Departmet of Electrical Engineering, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy (rafiei@ieee.org)

conditions. Unfortunately, it is still rather difficult to tune properly the parameters of PID controllers, because many industrial plants are often burdened with problems such as being of high order and existence of nonlinearities [18]. One of the first methods used as a classical tuning rule was proposed by Ziegler and Nichols. In general, it is often hard to determine optimal or near optimal PID parameters with the ZN method in many industrial plants [19] and [20]. For these reasons, it is highly desirable to increase the capabilities of PID controllers by adding new researches.

Genetic Algorithm (GA) [25]–[31] has recently received much interest for achieving high efficiency and searching globally optimal solution in search space. Due to its high potential for global optimization, GA has received great attention in control systems such as the search of optimal PID controller parameters. Although GA has widely been applied to many control systems, its natural genetic operations would still result in enormous computational efforts [26]-[27]. Though the GA methods have been employed successfully to solve complex optimization problems, recent research has identified some deficiencies in GA performance. This degradation in efficiency is more apparent in applications with the parameters being optimized are highly correlated. So, the crossover and mutation operations cannot ensure better fitness of offspring, because chromosomes of the population have similar structures and their average fitness is high toward the end of the evolutionary process [31]. Moreover, the poor premature convergence of GA degrades its performance and reduces its search capability [31].

To explore the superiority of the proposed optimization approach, the EDARLA method has been applied to design a PID controller for an Automatic Voltage Regulator (AVR) system for power generation system as an important industrial plant [21]-[24]. The generator excitation system regulates the generator's voltage and controls the reactive power flow using an AVR system. The role of the AVR is to hold the terminal voltage magnitude of a synchronous generator at a pre-specified level. Hence, the performance and stability of the AVR system seriously affect the security of the whole power system. In this paper, besides demonstrating how to employ the classic CARLA and DARLA as well as the proposed EDARLA methods to obtain the optimal PID controller parameters, it is shown that the proposed method has better performance compared to the conventional reinforcement learning methods. The paper is organized as follows. In section II, the conventional DARLA methods are briefly introduced. The proposed method is presented in section III. Section IV appropriately introduces the AVR system under study which is based on IEEE std 421.5 standards [21]. The extensive simulation results and comparisons are given in section V. Section VI concludes the paper.

II. CARLA and DARLA Techniques

In this section, both DARLA techniques are briefly reviewed [32]-[33].

A. DARLA

In order to implement DARLA, each DPF $f_i(d)$ must be stored and updated at discrete sample points. The most efficient data storage can be achieved using equal inter-sample probability rather than equal sampling at $d = 1,2,3,....N$, but this imposes some more computational burden. Like CARLA, any action set that produces an improvement in the system performance receives a higher-performance score $\beta(k)$, and thus its probability of reselection is increased. This is achieved by modifying $f_i(d)$ through the use of an Inverse Exponential Neighborhood function centered at the recent successful action. The neighborhood function increases the probability of the original action and the probability of all actions 'close ' to that selected as well. The assumption is that the performance surface over a range in each action is discrete and slowly varying. Within each iteration k of the algorithm, each action is chosen based on the corresponding probability distribution function $f_i(d,k)$ which is initially chosen to be a uniform function:

$$f_i(d,1) = \begin{cases} 1/N_i & where\ d = 1,2,...N_i, \\ 0 & otherwis \end{cases} \quad \textbf{(1)}$$

The action d is selected by solving:

$$\sum_d f_i(d) = Cumulative \quad \textbf{(2)}$$

Where the constant '*Cumulative*' is uniformly selected at random within the range [0,1]. When all *n* actions are selected, the set is evaluated in the environment for a suitable time, and the scalar cost $J_{cal}(k)$is calculated. Performance evaluation is then carried out using (9) and (10):

$$\beta(k) = \min\left\{\max\left\{0, \frac{J_{med} - J(k)}{J_{med} - J_{\min}}\right\}, 1\right\} \quad \textbf{(3)}$$

Again, the algorithm uses a reward/inaction rule. The action sets generating a cost not better than the current average level receive no reward ($\beta(k) = 0$), and the maximum reinforcement (reward) is also capped at $\beta(k) = 1$. After performance evaluation, each discrete probability function is updated according to the following rule:

$$f_i(d,k+1) = \begin{cases} \alpha(k)[f_i(d_i,k) + \beta(k)Q(d_i,r)] & if\ \ d_i = 1,2,..N; i = 1,2,..n \\ 0 & otherwise \end{cases}$$

(4)

Where Q is a symmetric *Inverse Exponential neighborhood* function centered on the action choice, *r =d_i(k), :*

$$Q(d_i, r) = \lambda 2^{-(d_i - r)^2} \quad \textbf{(5)}$$

And λ is an adjustable parameter that influences speed and resolution of the learning. The parameter $\alpha(k)$ is also calculated by (12) to renormalize the distribution functions in *k+1*-th iteration:

$$\alpha(k) = \frac{1}{\sum_{i=1}^{N_i} f_i(d_i,k) + \beta(k)Q(d_i,r)} \qquad \textbf{(6)}$$

A typical layout of the learning system by DARLA is shown in Fig. 1.

I. III. The Proposed Extended DARLA Method

Let '*n*' be the number of parameters must be adjusted so that the PI function reaches its minimum value. We can search for the controller's parameters in a for each cell dimensional space using a common discrete probability function (CDPF) $f_{X1,X2,\ldots\ldots\ldots,Xn}(x_1,x_2,\ldots\ldots,x_n)$ for each cell rather than '*n*' separate DPFs. The idea behind this strategy is that a CDPF has by far much more information than '*n*' separate DPFs. Each cell in the n-dimensional space must be stored and updated at discrete sample points which are here updated to either a new value or zero for simplicity. A typical layout for the proposed method is shown in Fig .2.

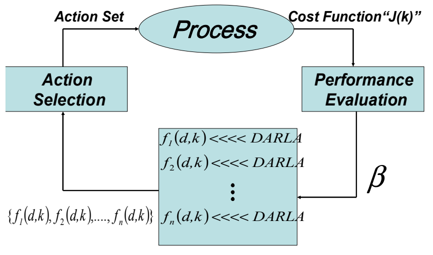
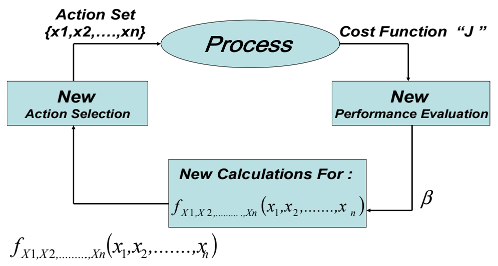


Fig . 1: Learning system by DARLA



Action Set {x1,x2,….,xn}
Process
Cost Function "J"
New Action Selection
New Performance Evaluation
New Calculations For :
$f_{X1,X2,\ldots\ldots\ldots,Xn}(x_1,x_2,\ldots\ldots,x_n)$
β
$f_{X1,X2,\ldots\ldots\ldots,Xn}(x_1,x_2,\ldots\ldots,x_n)$

Fig 2 Learning system by Proposed Algorithm

In this method, instead of '*n*' times calculating DPFs, we calculate one matrix function so that the speed of convergence increases by efficient matrix calculation algorithms. More importantly, it also potentially improves chance of reaching some better results closer to the globally optimal results. Suppose n=*3* then each three–dimensional probability function forms a cubic space as shown in Figure 4. The number of distinct values at each dimension is denoted as *Ndiv1, Ndiv2,* and *Ndiv3* respectively.

Within each iteration , each action has an associated probability density function $f_X(x)$ being used as the basis for its selection. The action sets improving the system performance, receive a higher-performance score, thus their probability of reselection increases through the learning sub-system. This is achieved by modifying $f_X(x)$ through the use of a three-dimensional neighborhood function centered at the successful action. The neighborhood function increases the probability of the original action, as well as the probability of the actions close to the point selected. With all *n* actions selected, the set is now evaluated in the environment for a suitable time, and a scalar cost value $J_{cal}(k)$ is calculated and compared with a memory set of previous values as described before.

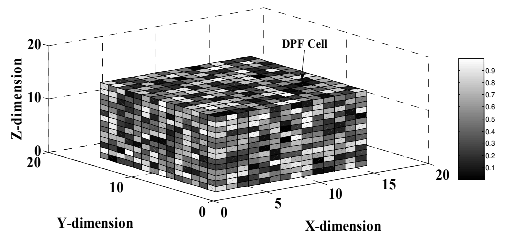


Fig. 3 A typical three- dimensional probability functional cube

After performing the performance evaluation process, each probability density function is updated according to a specified rule. Furthermore, EDARLA can consider several loops for searching. For example, if the number of loops is two, then the optimization is done through two stages (loops). In the first stage, the best subsection of the search space is found, then in the second stage, the algorithm proceeds searching for the best results within the determined subsection. This way, the algorithm would be much faster and less sensitive to the size of the initial search space defined as will be proved by various simulation results. The flowchart of the proposed reinforcement learning method is given by Fig. 5.

EDARLA Algorithm:

Step 1: Initialize the number of design variables (*Npar*) depending on the system under study, the number of divisions for each parameter (*Ndiv)* which, the number of iterations *(Niter,)* and finally the number of loops *(Nloop)* which determines the number of subsections.

Step 2: Start the loop.

Step 3: Generate a *Npar*-dimensional space and then set a *Npar*-dimensional uniform CDPF matrix which has $(Ndiv)^{Npar}$ cells and each cell has its own CDPF which is initially the same for all cells, but it must be updated in each iteration. The calculation of CDPF is very important, as the best cell has the highest CDPF.

Step 4: Start the iterations for the current loop which of course depends on its number of iterations specified and the number of loops.

Step 5: Select at random a cell with respect to its CDPF from the cubic structure shown so that the selection probability is proportional to its *CDPF*. It must be noted that there are different ways for this selection such as *roulette wheel selection* method used in this paper [34]. This step is performed according to the following three sub-steps:

a) Suppose a circle that has N_{div} sectors. Which each sector of the circle is respect to a cell. Angle of each sector calculates by:

$$\theta_i = \frac{360}{N_{div}} \quad , \quad i = 1,2,\ldots,N_{div} \qquad (7)$$

Now, select sector θ_{rand} at random (According to the DPFs) within $[0^0 \; , 360^0]$:

$$\theta_{rand} = rand(0,1) * 360 \qquad (8)$$

c) If $\theta_i = \theta_{rand}$ for some 'i', then sector i in the original search space is selected.

Step 6: First, the index of the selected cell is obtains. Then, this index is applied to equation (15) and N_{par} parameters (N_{par} actions) are obtained, the set is evaluated in the environment (control system) , and the scalar cost $J_{cal}(k)$ is calculated.

$$P(i) = E(i) * l(i) + G(i) \quad , \quad i = 1,2,\ldots,N_{par} \quad (9)$$

Where P(i), l(i), and G(i) are ith parameter or action, length of *i*th interval and beginning of *i*th interval, respectively.

Step *7: Evaluate the cost function according to the performance index of the system under control such as time and rate of error, control input, overshoot, steady-state error, etc. There are some coefficients which are the weighting elements. After the transient response, using a record of the system response, calculate the cost function (*$J_{cal}(k)$*) which is minimum for the best cell. The procedure compares the calculated cost function of the selected cell with the minimum of the cost function. It must be noted that in the first iteration, the minimum performance index is the same recently calculated performance index. The parameter* $\beta(k)$ *is updated according to the following rule in which 'k ' is the number of iteration:*

$$\beta(k) = \begin{Bmatrix} 1 & if & Jcal < J\min \\ 0 & if & Jcal > J\min \end{Bmatrix} \qquad (10)$$

If calculated cost function is improved ($J_{cal}(k) < J_{\min}$), then go to step 8, else set the CD*PF* of the selected cell to zero, and then go to step 9. This modification in updating the DPFs is take according to the suggestions for future works given in [12] for classic DARLA. This modification has improved the efficiency of the EDARLA.

Step 8: Take the calculated cost function *(* $J_{cal}(k)$*)* as the minimum cost function *(J$_{min}$)* , selected cell as the *New-Group,* and update the CD*PF* matrix as described below:

$$f(k+1) = f(k) + \beta(k)\,H(k)$$
$$H(k) = 2^{\left(-\frac{\| Indx\_Cell - New\_Group(k) \|^2}{N_{div}}\right)} \qquad (11)$$

Where, *Indx_Cell* is the index of each cell and New_Group (k) is the current group's center . One can compare the procedure with the original DARLA (5). The same normalization task is performed similar to (6).

Step 9:

If the iteration is finished, go to the step 2 for another loop and repeat the process to the *Final-Group*, else go to step 5.

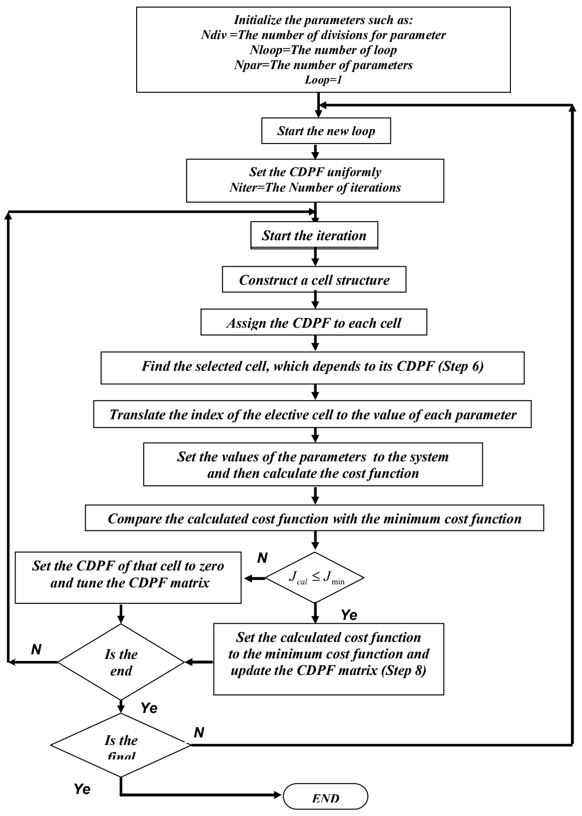

Initialize the parameters such as:
Ndiv =The number of divisions for parameter
Nloop=The number of loop
Npar=The number of parameters
Loop=1
Start the new loop
Set the CDPF uniformly
Niter=The Number of iterations
Start the iteration
Construct a cell structure
Assign the CDPF to each cell
Find the selected cell, which depends to its CDPF (Step 6)
Translate the index of the elective cell to the value of each parameter
Set the values of the parameters to the system and then calculate the cost function
Compare the calculated cost function with the minimum cost function
N
$J_{cal} \le J_{\min}$
Set the CDPF of that cell to zero and tune the CDPF matrix
Ye
Set the calculated cost function to the minimum cost function and update the CDPF matrix (Step 8)
N
Is the end
Ye
N
Is the final
Ye
END

Fig. 4 Flowchart of the proposed *EDARLA*

IV. PROPOSED OPTIMAL PID CONTROLLER FOR AVR SYSTEM

So far, PID controllers have widely been used in process control. With simple structure, they yet can effectively control various large industrial processes. There are many tuning approaches for these controllers, but each has own disadvantages or limitations. As a result, the design of PID controllers still remains a remarkable challenge for researchers. In simple words, the PID controller is used to improve the dynamic response as well as reduce or eliminate the steady-state error. The derivative term normally adds a finite zero to the open loop plant transfer function and can improve the transient response in most cases. The integral term adds a pole at origin resulting in increasing the system type and therefore reducing the steady-state error. Furthermore, this controller is often regarded as an almost robust controller. As a result, they may also control uncertain processes. The well-known PID controller transfer function is as follows:

$$C(s) = k_p + \frac{k_i}{s} + k_d s \qquad \textbf{(12)}$$

One of the important approaches used to design and tune the PID controllers including those are being used in AVR systems is the well-known Ziegler and Nichols (ZN) approach [20]. ZN is an well popular and interesting approach originated by Ziegler and Nichols in 1942 [20] and later extended in 1984 by Astrdm and Hagglund [19] . In this paper the proposed EDARLA is used as an automatic technique for optimal designing the PID parameters for a practical high order AVR system. The design method is fast and. The application results and comparisons with DARLA and ZN methods are given in the next section.

A. AVR System Under Study

The responsibility of an AVR is to hold the terminal voltage of a synchronous generator at a specified level. Hence, the performance and stability of the AVR seriously affects the security of the power system. In this paper, the AVR system under study has been modeled based on IEEE standard 421.5 [21]- [24]. The model takes into account all the major time constants and saturation effect and other nonlinearities. The block diagram of the AVR system with the PID controller is shown in Fig. 5. [24]. The figure does not show the saturation effects, but they are fully considered in all the design steps and simulations.

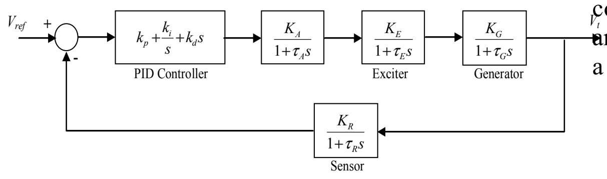


Fig.5- Block diagram of the AVR System with PID controller

B. Performance Index:

Generally, in many traditional optimal PID controller design approaches, some well-known performance indexes or performance criteria such as the integrated absolute error (IAE), the integral of squared-error (ISE), or the integrated of time-weighted- squared-error (ITSE) are widely used. Each of the three integral performance criteria has its own features [19]. Despite the existence of some classic performance indexes addressed, a more effective performance criterion in time domain is here suggested for designing the PID controller. The new performance criterion J is defined as follows and considers some broader requirements in more explicit manner where e(t) is the error between the desired and real output quantities:

$$\min_k J(K) = G_e \int_0^T e^2(t)dt + G_u \int_0^T u_c^2(t)dt + G_M M_P + G_s E_{ss} + G_d \sup\left|\frac{de(t)}{d(t)}\right|$$

(13)

Where T is the total simulation time and it is $T=20\ sec$ for this study , $e(t)$ is the tracking error , $u_c(t)$ is the control input , M_p is the amount of the overshoot , E_{ss} is the steady-state error at t=T, and G- coefficients are the weighting elements (G_e=10 ,G_u=1 ,G_M=10 , G_s=10 , G_d=5). The parameters of the PID controller are calculated based on the following approaches and the results are compared:

a) Conventional DARLA
b) Extended DARLA (EDARLA)
c) Conventional Ziegler-Nichols [24]
d) Genetic Algorithm (GA) [18]

V. Simulation Results

A. Performance of the EDARLA

For the practical AVR system under study, the results show that the proposed method can perform an excellent search for the optimal PID controller parameters quickly. As will be shown through extensive simulation studies, one of drawbacks of the DARLA is that it is quite sensitive to the initial ranges considered for the design variables forming the overall search space. The proposed method is more efficient as well as more robust against the search space volume. For this comparative study, fortunately, the normal ranges of three controller parameters that is *K_p, K_i and K_d* considered in other references [22]- [24] are available. The AVR system parameters are listed in Table 1 [22]. The AVR parameters and some other symbols used in this study are shown in Tables 1 and 2. Fig. 6 shows the original step response of the AVR system without controller. In this case, M_p%=50.51, E_{ss}%=9.06, T_r=0.273s and T_s=5.565s. As seen, the results are not satisfactory for a practical system.

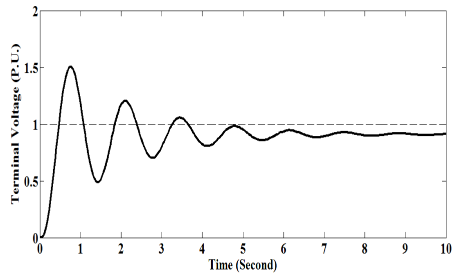


Fig. 6. Step response of the AVR system without controller

Table 1 . The AVR system parameters

Description	Parameter	Value
Amplifier Gain	K_A	10
Amplifier Time Constant	τ_A	0.1
Exciter Gain	K_E	1
Exciter Time Gain	τ_E	0.4
Generator Gain	K_G	1
Generator Time Constant	τ_G	1
Sensor Gain	K_R	1
Sensor Time Constant	τ_R	0.01

Table 2 . The used symbols

Variable Name	Description
Loop-k	*k-th* of loop (*k*=1,2,3)
F-Iter-k	Number of iteration in loop *k* (*k*=1,2,3)
J_{min}	Minimum value of Performance Index
Ts	Settling time
Tr	Rise time
$M_P\%$	Percent of overshoot
Ess%	Percent of steady state error

Table 3 . The Results of Simulation for EDARLA)

case	Number of Iterations	Loop–1	Loop-2	Loop-3	Design time (s)	J_{min}	M_P%	Tr(s)	Ts(s)	Ess%	K_p	K_i	K_d
1	10	10	0	0	0.272	17.44	19.45	0.509	2.85	0.00	0.600	0.600	0.200
2	20	20	0	0	0.561	14.39	7.44	0.396	1.135	0.07	0.900	0.400	0.200
3	20	10	10	0	0.535	13.73	0.25	1.146	1.691	0.08	0.744	0.160	0.344
4	50	20	15	15	1.386	13.37	0	0.518	0.962	0.00	1.044	0.240	0.296
5	100	50	25	25	2.856	10.02	0	0.854	1.036	0.00	0.636	0.176	0.224
6	150	50	50	50	4.320	8.980	0	0.692	1.087	0.00	0.574	0.176	0.144
7	200	100	50	50	5.467	8.42	0	0.433	0.673	0.00	0.948	0.256	0.232

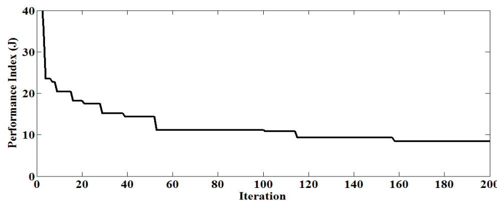


Fig. 7. Convergence tendency of the EDARLA (case 7)

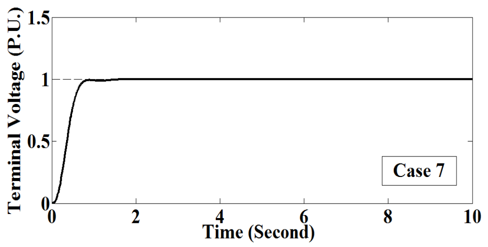


Fig. 8. Terminal voltage step response of the AVR system with optimal PID controller (EDARLA)

In this section, the parameters of the PID controller are calculated by the proposed EDARLA. Considering the results already reported in [15], the ranges of the three parameters K_p, K_i *and* K_d are taken as [0 1.5], [0 1], and [0. 1] respectively. For calculation of the PID parameters, each interval is divided into 5 slots in the first, second, and three loops. The simulation results of the EDARLA for different number of loops and different number of total iterations are summarized in Table 3. It can be seen that the final results are quite interesting, and excellent results can be obtained through even less than 50 iterations. Moreover, comparing cases 2 and 3 (with the same number or iterations equal 20) reveals that having two loops rather than one loop can clearly result in better results as reflected in the final cost functions of the addressed case studies. Also, Figs. 7 and 8 show the convergence characteristics of the proposed method and terminal voltage step response of the AVR system respectively for different simulation conditions. As seen, through about 50 iterations, the EDARLA method successfully converges and provides good performance. The results prove that the proposed method (EDARLA) can search for optimal PID controller parameters quickly and efficiently

B. Comparison between the Proposed EDARLA, DARLA and ZN Methods

figures 9 - 11 compare the results ZN , DARLA, and EDARLA methods. The EDARLA has totally run for 50 iterations only while the DARLA has trained for 200 iterations , and as seen EDARLA provides much better results. Also, Fig. 12 shows the convergence curves and step response of the AVR for both DARLA and EDARLA methods after 200 iterations. Again, one can clearly see that the EDARLA has provided much better results with faster convergence. Also, The PID controller parameters calculated by Ziegler-Nichols method [4,25] are as follow:

$$K_p = 1.02 \quad K_i = 0.31 \quad K_d = 0.1847 \qquad (14)$$

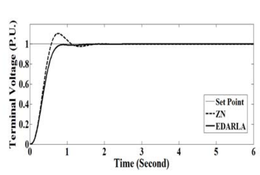


Fig 9. Comparison of ZN and EDARLA methods

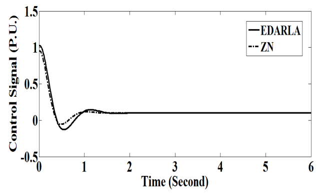


Fig 10. The Control Signal of ZN and EDARLA methods

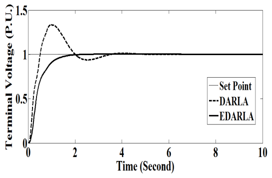


Fig 12. The responses of

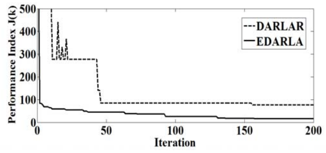


Fig 11. performance index of DARLA and EDARLA

DARLA and EDARLA methods

C. Robustness to the Design Variables Ranges

As Already addressed, one of key issues in both DARLA and CARLA methods is the range of each design variable need to be pre-defined. This is not a trivial task, as the designer cannot always guess some appropriate ranges. If the ranges are too small, that makes the overall search space too small that the procedure may fail to find acceptable optimal points. On the other side, large ranges can make the algorithm too time consuming. Even worth, for a fixed number of slots (Ndiv), larger ranges may lead to even worth results as shown in Tables 4 and 5. The tables show the comparative results of the conventional DARLA and the proposed EDARLA. Both methods have run for 200 iterations. As the results show, the DARLA fails to reach good results even for small ranges. Moreover, as the ranges increase, the overall performance index increases. The changes in the performance index, rise time, overshoot, and overshoot is notably less for the EDARLA. This feature is quite important, because as already mentioned, in most cases, the best ranges for the search space cannot be determined exactly. Hence, a need for a heuristic approach such as EDARLA is apparent. The Results prove the good robustness of the EDARLA and explain how the conventional DARLA is too sensitive to the ranges taken for its variable. So, the most efficient, robust, and optimal results can be gained by the proposed method (EDARLA). In the next subsections performance, efficiency, and sensitivity of GA and CARLA approaches are also investigated.

Table 4 . Comparison of proposed method (EDARLA) and conventional DARLA run for 200 iterations

Interval of Parameters	Conventional DARLA($\lambda = 1$)		EDARLA (Proposed Method)	
	J_{min}	Simulation time (s)	J_{min}	Simulation time (s)
[0 1.5]	9.762	7.315	8.42	5.467
[0 5]	15.798	8.323	10.66	7.56
[0 10]	26.737	16.848	13.38	16.555
[0 15]	40.575	22.06	14.03	16.928
[0 20]	76.424	23.860	15.85	18.792

Table 5. Effect of the search space: A comparison between the proposed EDARLA and the conventional DARLA

Interval of	CARLA			EDARLA		
Parameter	M_P%	Ts(s)	Ess%	M_P%	Ts(s)	Ess%
[0 1.5]	0	2.22	0.71	0	0.433	0.67
[0 5]	4.21	0.77	0.18	0	1.33	0.0
[0 10]	6.32	2.3	0.15	0	1.07	0.12
[0 15]	22.15	2.135	0	0.346	1.336	0.11
[0 20]	33.19	2.85	0	0.15	1.234	0.0

D. Comparison between the Proposed EDARLA and GA Methods

In order to more highlight the advantages of the proposed method, we also implemented the GA (Genetic Algorithm) method [18] and [22]. The characteristics of the two controllers using the same performance index as defined by (22) are compared. The GA parameters are as follow:
Population Size: 25
Crossover Rate (Pc) = 0.75
Mutation Rate (Pm) = 0.0075

Table 6 shows the GA approach results for different ranges of the parameters. As seen the GA does not provide good results even over 200 populations. It is also more sensitive to the variables ranges. Figs 13-15 compare the step responses of the AVR system when GA and EDARLA methods are used for training the PID controller for some different ranges of the parameters. As seen, EDARLA provides the best results in comparison with the results of the other methods discussed. Also, Fig 16 compares the convergence curve of GA and EDARLA methods. It must be noted that while both methods are executed for 200 iterations, the EDARLA needs 200

Table 6. Simulation Results for designing PID controller by GA : Sensitivity Analysis

case	Interval of Parameter	simulation time (s)	J_{min}	M_P%	Tr(s)	Ts(s)	Ess%
1	[0 1.5]	128.96	10.11	5.78	0.962	3.21	0.0
2	[0 5]	155.04	14.126	13.64	1.026	4.76	0.0
3	[0 10]	346.1	25.613	15.97	0.753	4.19	0.03
4	[0 15]	366.72	34.19	17.74	1.036	5.68	0.5
5	[0 20]	365.57	47.99	36.568	1.742	12.24	0.0

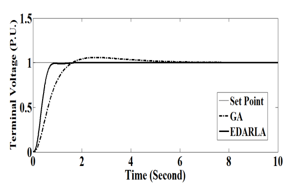


Fig. 13 The responses of GA and EDARLA methods (range of parameters : [0 1.5])

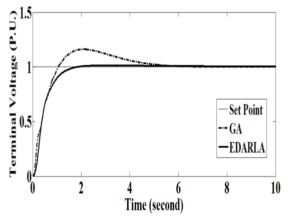


Fig. 14 The responses of GA and EDARLA methods (range of parameters : [0 10])

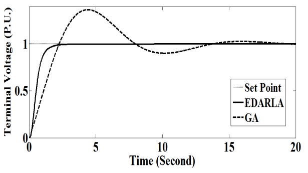


Fig. 15 The responses of GA and EDARLA methods (range of parameters : [0 20])

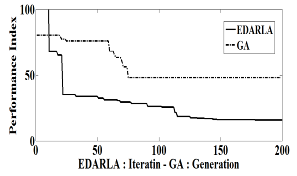


Fig. 16 Performance index of GA and EDARLA methods (range of parameters : [0 20])

(Number of Iterations for EDARLA and Generations for GA : 200)

fitness function evaluations, but the GA needs 200 ×25 function evaluations, because each population has 25 elements. That makes the GA too time consuming in comparison to DARLA and EDARLA.

VI. Conclusion

This paper introduces a new intelligent method for optimizing the parameters of the PID controller for an AVR system. The proposed method is an extended version of DARLA method called EDARLA which optimizes the controller parameters while the variables are not considered independent opposed to the classic DARLA. By using matrix calculation, the speed of convergence can be increased and the system can be used for many real time applications. Superior performance, robustness, and efficiency of the proposed method have been proved through extensive simulation results including comparisons with GA, Ziegler-Nichols, and DARLA methods. The extensive studies carried out clearly proves that the proposed approach is an excellent candidate for optimizing various control problems including AVR system thank to its high efficiency, speed, and robustness.

REFERENCES

[1] S.Y. Oh, J.H. Lee, D.H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following", IEEE Trans. on Vehicular Technology, Vol. 49, Issue 3, May 2000, pp.997 – 1005.

[2] A. A. Harati, M.N. Ahmadabadi, and B.N. Araabi, " Knowledge based multiagent credit assignment: A study on task type and critic information," Intelligent Automation and Soft Computing 2007 (AutoSoft), vol. 13, no. 3.

[3] A. Imanipour, M.N. Ahmadabadi, B.N. Araabi, M. Asadpour, and R. Siegwart, "Knowledge-based extraction of area of expertise for cooperation in learning," in Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst. (IROS), Beijing, China, 2006, pp.3700–3705.

[4] D. Vengerov, N. Bambos, and H.R. Berenji, "A Fuzzy Reinforcement Learning Approach to Power Control in Wireless Transmitters," IEEE Transactions On Systems , Man, And Cybernetics-PartB:Cybernetics,vol.35,No.4., 2005.

[5]G. Tesauro, "Reinforcement Learning in Autonomic Computing ",Published by the IEEE Computer Society , IEEE INTERNET COMPUTING, 2007, pp.22-30.

[6] B.N. Araabi, S. Mastoureshgh, and M.N Ahmadabadi, " A Study on Expertise of Agents and Its Effects on Cooperative Q-Learning," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 37, no. 2, 2007, pp. 398–409.

[7] M. N. Howell, G. P. Frost, T. J.Gordon and Q. H. Wu, "Continuous Action Reinforcement Learning Applied to Vehicle Suspension Control," Mechatronics, pp. 263-276, 1997.

[8] G.P. Frost, "Stochastic optimization of vehicle suspension control systems via learning automata, " Ph.D. Dissertation, Aeronautical and Automotive Engineering Department, Loughborough University, 1998.

[9] .N. Howell, M.C. Best, "On-Line PID Tuning For Engine Idle-Speed Control Using Continuous Action Reinforcement Learning Automata", Control Engineering Practice 8 147-154, Elsevier Science Ltd., 2000.

[10] M.N. Howell, T.J. Gordon, "Continuous Action Reinforcement Learning Automata And Their Application To Adaptive Digital filter Design", Engineering Applications of Artificial Intelligence 14, Elsevier Science Ltd., pp.549–556, 2001.

[11] G. Tesauro et al., "A Hybrid Reinforcement Learning Approach to Autonomic Resource Allocation, " Proc. Int. Conf. Autonomic Computing (ICAC), IEEE CS Press, 2006, pp. 65–3.

[12] V.C. Zvacek, Magnetic Bearing Controller Tuning Through CARLA Learning Method", Thesis presents to university of applied sciences, Merseburg, Aug. 2004.

[13] T. Fukuda, Y. Hasegawa, K. Shimojima, F. Saito, "Reinforcement learning method for generating fuzzy controller", IEEE International Conf. on Evolutionary Computation, Vol. 1, 29 Nov.-1 Dec.,1995 , pp.273 -278.

[14] T. Fukuda, Y. Hasegawa, K. Shimojima, F. Saito, "Self scaling reinforcement learning for fuzzy logic controller" IEEE International Conf. on Evolutionary Computation, Vol., 20-22 May, 1996 pp. 247 – 252.

[15] M. Kashki, A.A. Gharaveisi, F. Kharaman, "Application of CDCARLA Technique in Designing Takagi- Sugeno Fuzzy Logic Power System Stabilizer (PSS)", IEEE International Conf. on Power and Energy, PECon '06, 28-29 Nov. 2006, pp. 280-285.

[16] M.N. Howell, T.J. Gordon, and M.C. Best, " The Application of Continuous Action Reinforcement Learning Automata to Adaptive PID Tuning, " Proceedings of IEE Control Seminar on Learning Systems for Control, 2000, pp. 1-10, Birmingham, UK.

[17] T. L. Seng, M. B. Khalid, and R. Yusof, "Tuning of a neuro-fuzzy controller by genetic algorithm," IEEE Trans. Syst., Man, Cybern. B, vol. ,29 pp. 226–236, Apr. 1999.

[18] R.A. Krohling , J.P. Rey, "Design of optimal disturbance rejection PID controllers using genetic algorithms", IEEE Trans .on Evolutionary Computation, Volume 5, Issue 1, Feb 2001, pp.78 – 82.

[19] K.J. Astron, and T. Hagglund, " PID controllers: Theory, design, and tuning." Research Triangle Park, NC, USA, 1995.

[20] J.G. Ziegler, and N.B. Nichols, " Optimum setting for Automatic Controllers, " Transactions of ASME, 1942, VOL 64, pp. 759-768.

[21] IEEE Recommended Practice for Excitation System Models for Power System Stability Studies, IEEE Standard 421.5-2005.

[22] Z. L. Gaing, "A Particle Swarm Optimization Approach For Optimum Design of PID Controller in AVR System", IEEE Transaction on Energy Conversion, vol. 19, No. 2, pp. 384-391, June 2004.

[23] H. Yoshida, K. Kawata, and Y. Fukuyama, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," IEEE Trans. Power Syst., vol. 15, pp. 1232– 1239, Nov. 2000.

[24] F. Naderi, A.A. Gharaveisi and M. Rasidinejad, **"**Optimal Design of Type_1 TSK Fuzzy Controller Using GRLA for AVR System**"** , 2007 Large Engineering Systems Conference on Power Engineering, Canada, 10-12 Oct. 2007, pp. 106-111.

[25] R. A. Krohling, H. Jaschek, and J. P. Rey, "Designing PI/PID controller for a motion control system based on genetic algorithm," in Proc. 12th IEEE Int. Symp. Intell. Contr., Istanbul, Turkey, July 1997, pp. 125–130.

[26] C.H. Chou, "Genetic Algorithm-Based Optimal Fuzzy Controller Design in the Linguistic Space", IEEE Trans. On Fuzzy Systems, vol. 14, No. 3, pp. 372-385, June 2006.

[27] C.H. Wang, T.P. Hong, and S.S. Tseng, "Integrating fuzzy knowledge by genetic algorithms," IEEE Transaction on Evolutionary Computation**,** vol. 2, no. 4, pp. 138-149, 1998.

[28] G.K. Kau, S.K. Mukherjee, C.K. Loo, and L.C. Kwek, "Evolutionary PID control of non-minimum phase plants", 7th International Conf. on Control, Automation, Robotics and Vision, 2002, ICARCV, Vol. 3,Issue , 2-5 Dec. 2002 , pp.1487-1492.

[29] L. Sanchez, 'Interval-valued GA-P algorithms', IEEE Trans. on Evolutionary Computation, Vol. 4, Issue 1, Apr 2000, pp. 64 – 72.

[30] M. Russo, "Genetic fuzzy learning" IEEE Trans. on Evolutionary Computation, Vol. 4, Issue 3, Sep. 2000 pp.259 – 273.

[31] D. B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 2 ed. New York: IEEE Press, 2000.

[32] J. Kabudian, M.R. Meybodi, M.M. Homayounpour, "Applying continuous action reinforcement learning automata(CARLA) to global training of hidden Markov models", International Conf. on Coding and Computing, ITCC2004, Vol. 2, pp. 638-642.

[33] M. Aleksy, A. Korthaus, M. Schader, "CARLA - a CORBA-based Architecture for Lightweight Agents", IEEE/WIC International Conf. on Intelligent Agent Technology ,IAT2003,13-16 Oct. 2003 pp.111 - 118 .

[34] B. Liu, 'Uncertain Programming', Book, Second Edition, UTLAB, 2008, http://orsc.edu.cn/liu/up.pdf