Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2013) 21: 1603 – 1614
© TÜBİTAK
doi:10.3906/elk-1109-40

Opposition-based discrete action reinforcement learning automata algorithm case study: optimal design of a PID controller

Fatemeh MOHSENI POUR,*[*] Ali Akbar GHARAVEISI
Department of Electrical Engineering, Faculty of Engineering, Shahid Bahonar University of Kerman, Kerman, Iran

Received: 20.09.2011 • **Accepted:** 09.04.2012 • **Published Online:** 02.10.2013 • **Printed:** 28.10.2013

Abstract: In this paper, the discrete action reinforcement learning automata (DARLA) method is expressed. The performance of the reinforcement learning algorithm is improved using the opposite concepts. This is an automatic method that can find the global optima without any knowledge about the parameters of the research space. To find the global optimal point, the interval that contains the optima is determined by DARLA as the cost function is minimized. In the opposition-based DARLA method, learning is performed based on opposition. The main idea in the opposition is to consider the search direction and its opposite at the same time to reach the candidate solution. This concept has increased the convergence rate and accuracy, and this algorithm can be used for many real-time applications. To prove this, the opposition-based DARLA is proposed to design a proportional-integral-derivative (PID) controller for the automatic voltage regulator system. The experimental results for the optimizing PID controller problem demonstrate the superior performance of the proposed approach.

Key words: Opposite concepts, discrete action reinforcement learning automata algorithm, automatic voltage regulator system

1. Introduction

The reinforcement learning method is a computational method for goal-directed and decision-making problems. The proposed method is different from other intelligent computational methods because it relies on the interaction with the environment, considering uncertainty and a nonexact model of the environment.

This process is performed by a reinforcement learning agent, in a framework that consists of 3 elements: state, decision, and reward. Although this framework is very simple, it covers a wide range of artificial intelligence problems. This method is based on the action–reaction concept while considering uncertainty. To increase the probability of finding better points, an automaton based on the probability density functions and manipulating them is applied [1,2].

In this paper, the discrete action reinforcement learning automata (DARLA) method is expressed. This method has a main disadvantage; it is a single-point search algorithm, so the convergence rate is not good for real-time problems. In this paper, the opposition concept is used to improve the accuracy and convergence rate.

In the simulation part, first, both DARLA and its improved version have been applied to some benchmark functions mentioned in the Appendix. According to this comparison, the development and improvement of this algorithm is discussed, and then the proposed method is used to design a proportional-integral-derivative (PID)

[*]Correspondence: fomopo1986@yahoo.co.uk

controller for an automatic voltage regulator (AVR) system. For a stable electrical power service, it is very important to make the response of the AVR of its synchronous generator very efficient and fast. Due to the simplicity and low cost of the analog PID, it is still used as the AVR controller. However, the parameters of the PID controller are not tuned easily. In this paper, the opposition-based DARLA method is proposed to find the optimum values of these parameters.

2. DARLA

The reinforcement learning automata method, was first presented [1,3,4] by Howell et al. in 1997 [1,5]. In this paper, the DARLA algorithm is used. This method is based on interaction with the environment and employs probability density functions to find the optimum value of the decision variables of the problem. The simplified mechanism of DARLA is that if a set of the decision variables improve the system's performance, the probability of choosing them must be increased. The decision variables in DARLA have discrete values. Thus, for finding the global optimal solution of a problem, the search space is divided into fine intervals. One point in each of these intervals is selected to represent the behavior of that interval and the performance of the system at that point is used to estimate the system's performance over all of the points of that interval. In this paper, the middle point of each interval is selected as the decision variable and the interval that contains the optima is determined by DARLA.

The algorithm steps:

1. Randomly select a set of decision variables according to their cumulative distribution functions.
2. Supply the selected set to the test function.
3. Calculate the output of the test function and calculate the cost function.
4. Calculate the reinforcement signal according to the cost function.
5. Change the probability density functions using the reinforcement signal.
6. Find the cumulative distribution functions by integrating the density functions and return to step 1.

The initial probability density functions are defined as [1–4]:

$$f_i^{(0)}(d) = \begin{cases} \frac{1}{N_i} & d = 1, 2, ..., N_i \\ 0 & other \end{cases} \quad ; \quad i = 1, 2, ..., n \tag{1}$$

where n is the number of decision variables and N_i is the number of intervals of the ith decision variable. When the algorithm starts, the new set of decision variables is selected in each step according to probability density functions and is fed into the test function. This selection is done using the cumulative probability of each variable according to Eq. (2) [1–4]:

$$C_i^{(k)}(d) = \sum_{q=1}^{d} f_i^{(k)}, \quad d = 1, 2, ..., N_i \tag{2}$$

where $C_i^{(k)}(x)$ is the cumulative probability of the ith decision variable in the kth iteration. After calculating the cumulative probability, a uniform random number $r \in (0, 1]$ is used to choose the decision variable or interval number d. Obviously, the variable value that has a higher corresponding probability density has a greater chance of being selected. After selecting the variable values in the previous step, these values are applied to the test function or plant, and then the cost function is calculated. The cost function structure is very important and has the main role in finding the solution of any optimization problem. Usually, the cost function is in the form of a weighted combination of performance indices, like Eq. (3) [1–4]:

$$J\left(k\right) = G_1 P_1\left(Y\right) + G_2 P_2\left(Y\right) + ... + G_m P_m\left(Y\right), \tag{3}$$

where G_i, P_i, and $Y = [y_1, y_2, ..., y_l]$ are the weighted coefficient, performance index, and output vector, respectively. In this paper, the test function is used as the cost function itself.

The DARLA structure is designed to minimize the cost function. In each iteration, when the cost function value is calculated, it will be compared with the cost value of the previous iterations, and then the reinforcement signal is calculated accordingly [1–4]. The reinforcement signal is the suitability criterion of the selected variable values, so when the set of the selected values is more suitable, the value of the proposed signal is larger. For calculating the reinforcement signal, Eq. (4) is used [1–4]:

$$\beta\left(J\right) = \min\left\{1, \max\left\{0, \frac{J_{mean} - J}{J_{mean} - J_{\min}}\right\}\right\}, \tag{4}$$

where J_{mean} and J_{min} are the average and minimum of the previous cost values. The range of the β variation is between 0 and 1. Cost values that are larger than the mean of the previous cost values lead to a zero reinforcing (0), while cost values that are smaller than the mean lead to a reinforcing value equal to 1 [1–5]. After calculating the reinforcement signal, the probability density functions are manipulated proportional to the reinforcement signal and the shapes of the probability density functions change.

One of the methods for changing the probability density function is to use the Gaussian function Q as in Eq. (5):

$$Q\left(d, r\right) = \lambda 2^{-(d-r)^2} \tag{5}$$

where the positive number λ is an effective parameter on the accuracy and convergence rate [1–4]. Probability density functions are updated by Eq. (6), where α is used for normalizing the probability density function and is defined as in Eq. (7) [1–4].

$$f_i^{(k+1)}\left(d\right) = \alpha_i^{(k)}\left(f_i^{(k)} + \beta\left(k\right) Q\left(d, \tilde{d}_i\right)\right); \quad d = 1, 2, ..., N_i; \quad i = 1, 2, ..., n \tag{6}$$

$$\alpha_i^{(k)} = \frac{1}{\sum\limits_{q=1}^{N_i} f_i^{(k)} + \beta\left(k\right) Q\left(d, \tilde{d}_i\right)} \tag{7}$$

After changing the probability density functions, the algorithm iterates. By repeating this cycle enough times, the probability density functions will be maximized at the optimal points.

Hence, by the last iteration, in which these functions reach their maximum value, the selected points are the optimal points. DARLA has a repetitive structure and a criterion is needed for terminating the algorithm, so in this paper, a ‘repeating algorithm for a certain number of iterations’ is used as the terminating condition [1–4].

3. Opposition concept

In most of the artificial intelligence algorithms, learning begins at an initial random point and moves toward the desired solutions available. If the initial guess selects a point not too far from the optimal solution, the algorithm converges to it quickly [6–8]. However, it is usual to begin with an initial guess much farther away from the optimal solution. In the worst case, the optimal solution is in the 'opposite location' of the initial guess, so the approximation, search, or optimization will take considerably more time or becomes intractable [6,7,9]. Of course, without any prior knowledge, it is not possible to make the best initial guess. Logically, if a search for the optimal solution is done in a specific direction, it must be done in the direction opposite to it simultaneously [6–9]. Thus, if we are looking for x, then calculating the opposite value $\tilde{x}$ is the first step [6]. It must be considered that opposition points are different from each other based on their calculation criteria. If the criterion of opposition is the 'central point', then all of the dimensions of a specific point will be opposed, and the result is the 'opposite point' or 'absolutely opposite point'. If the opposition is done according to an axis or a plane, then the result is called the 'semiopposite point', because at least one dimension remains unchanged and the result point is not a complete opposite point; in other words, it is only partially opposed.

3.1. Opposite number

If $x \in R$ is a real number defined on a certain interval, $x \in [a, b]$, then the opposite number $\tilde{x}$ is defined as follows [6,8,9]:

$$\tilde{x} = a + b - x. \tag{8}$$

For example, if $a = 0$ and $b = 0$, then the opposite number is:

$$\tilde{x} = -x. \tag{9}$$

Similarly, when the opposite number is in a multidimensional space, it can be defined also [6,8,9].

3.2. Opposite point

If $P(x_1, x_2, ..., x_n)$ is a point in an n-dimensional coordinate system with $x_1, x_2, ..., x_n \in R \quad \forall i \in \{1, 2, ..., n\}$, then the opposite point of P is completely defined as $\tilde{P}(\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$ by its components $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$ [6–8], where:

$$\tilde{x}_i = a_i + b_i - x_i \quad i = 1, ..., n \tag{10}$$

By this definition, all of the dimensions of a specific point are opposed. While the definition in Eq. (10) is the criterion of the opposition regarding the central point of a certain domain, the result point is called 'the opposite point' or 'the absolutely opposite point'.

3.3. Semiopposite point

The main idea in this method is to oppose according to an axis or a plane, so that at least one dimension remains unchanged and the result point is not completely on the opposite side. This point is called the 'semiopposite point'. If the proposed point $P(x_1, x_2, ..., x_n)$ has n dimensions and each dimension can take 2 states (x_i and $\tilde{x}_i$), then 2^n points can be obtained by various combinations of states. One of these points is the main point ($P(x_1, x_2, ..., x_n)$) and another is the opposite point($\tilde{P}(\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$), so the rest of the $2^n - 2$ points are all semiopposite points.

For example, for n = 2 and $P(1,0)$ as the main point, where $x_1 \in [0,1]$ and $x_2 \in [0,1]$, the set of opposite points (including the absolutely opposite point and 2 semiopposite points) are obtained as the following:

$$\begin{aligned}
\tilde{x}_1 &= a_1 + b_1 - x_1 = 0 + 1 - 1 = 0 \\
\tilde{x}_2 &= a_2 + b_2 - x_2 = 0 + 1 - 0 = 1 \\
&\left.\begin{aligned} \bar{P}(\tilde{x}_1, x_2) &= \bar{P}(0,0) \\ \bar{P}(x_1, \tilde{x}_2) &= \bar{P}(1,1) \end{aligned}\right\} : \textit{ semi-oppsite} \\
\tilde{P}(\tilde{x}_1, \tilde{x}_2) &= \tilde{P}(0,1) : \textit{ Absolutely opposite}
\end{aligned}$$

The opposition-based optimization can be defined using the set of opposite points.

3.4. Opposition-based optimization

Let $P(x_1, x_2, ..., x_n)$ be a candidate solution in an n-dimensional space, where $x_i \in [a_i, b_i]$,$\forall i \in \{1, 2, ..., n\}$ [6,8]. Let $f(x)$be the cost function used in finding the optimal solution [6,8,9]. According to the opposite definitions, $\tilde{P}(\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n)$(opposite to $P(x_1, x_2, ..., x_n)$) and the set of semiopposite points are calculated. Next, in each iteration, these points are applied to the cost function $f(.)$. Any point that has the smallest cost function is the fitness point regarded by all of the points (main point, absolutely opposite point, and semiopposite points), and the learning continues according to it [6–9].

The opposite definition helps the optimization algorithm to find more suitable places in the search space and to reach the solution faster. The comparison between the main point and its opposites increases the convergence rate by shortening the search intervals in each dimension of the search space. Considering the interval $[a_1, b_1]$in Figure 1 [6–9], the solution for a given problem can be found by a repeated examination of a primary guess and the counter-guess [6–9]. In Figure 1, the opposite number xo of the initial guess x is generated [6–9]. Based on whether the estimate or its opposite is closer to the solution, the search interval can be halved repeatedly until either an estimate or its opposite stand close enough to the solution [6–9].

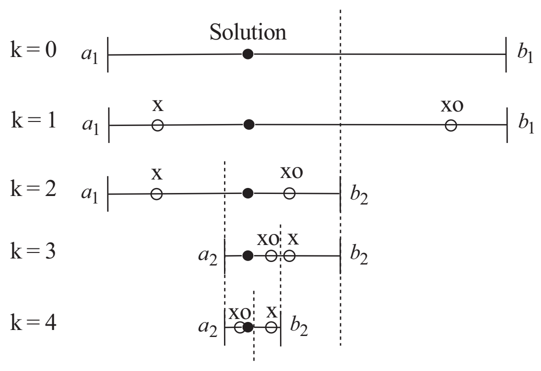
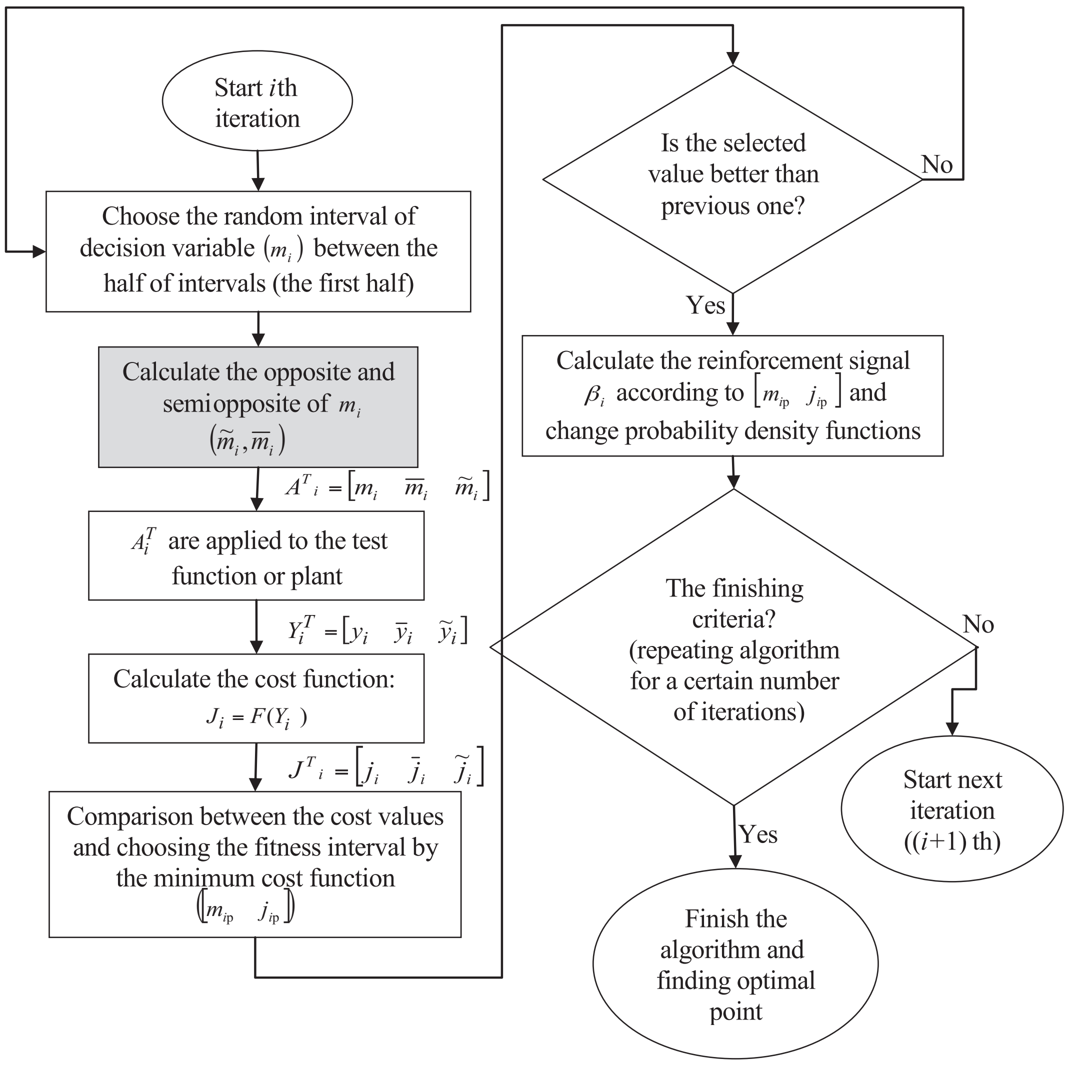


Figure 1. Solving a 1-dimensional equation via repeatedly halving the search interval with respect to the optimality of the estimate x or the opposite-estimate xo [6–9].

4. Opposition-based DARLA algorithm

Reinforcement learning automata algorithms are single-point search algorithms, so their convergence time is very long. By using a set of opposite points, the convergence rate is increased. We use this concept in the point generation part and the cost function evaluation part. In the opposition-based DARLA algorithm, instead of selecting just one point, a set of points containing the main point, absolutely opposite point, and semiopposite points is used for searching the problem space. The cost function is evaluated for each point of this set and the point that has the smallest cost value is chosen as the fitness point, and the algorithm continues based on it.

A flowchart of the opposition-based DARLA is shown in Figure 2. The operation related to the opposite concept is shown in the gray block.



Start ith iteration
Choose the random interval of decision variable (m_i) between the half of intervals (the first half)
Calculate the opposite and semiopposite of m_i $(\tilde{m}_i, \overline{m}_i)$
$A^T{}_i = [m_i \quad \overline{m}_i \quad \tilde{m}_i]$
A_i^T are applied to the test function or plant
$Y_i^T = [y_i \quad \overline{y}_i \quad \tilde{y}_i]$
Calculate the cost function: $J_i = F(Y_i)$
$J^T{}_i = [j_i \quad \overline{j}_i \quad \tilde{j}_i]$
Comparison between the cost values and choosing the fitness interval by the minimum cost function $([m_{ip} \quad j_{ip}])$
Is the selected value better than previous one?
No
Yes
Calculate the reinforcement signal β_i according to $[m_{ip} \quad j_{ip}]$ and change probability density functions
The finishing criteria? (repeating algorithm for a certain number of iterations)
No
Yes
Start next iteration ((i+1) th)
Finish the algorithm and finding optimal point

Figure 2. The flowchart of the opposition-based DARLA for the ith iteration.

5. Simulation

5.1. Benchmark functions

The proposed algorithm (DARLA) and its improved version (opposition-based DARLA) are applied to some benchmark functions. These test functions and their properties are summarized in Table 1 [10]. There are 2 unimodal $(F_1 - F_2)$ and 6 multimodal $(F_3 - F_8)$ functions, among which 4 of them are nonseparable (F_2, F_3, F_5, F_8) and 4 are separable (F_1, F_4, F_6, F_7) [10].

Table 1. The benchmark functions [10].

Function name	Function equation	n	Initial parameter bounds	Minimum value of the function
Sphere model	$f_1(x) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$	F_{1Min} *(0,0,…,0) = 0*
Hyper-ellipsoid	$f_2(x) = \sum_{i=1}^{n} 2^{i-1} x_i^2$	30	$[-100, 100]^n$	F_{2Min} *(0,0,…,0)=0*
Generalized Rosenbrock	$f_3(x) = \sum_{i=1}^{n-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	$[-30, 30]^n$	F_{3Min} *(1,1,…,1)=0*
De Jong, i.e. noise	$f_4(x) = \sum_{i=1}^{n} i x_i^4 + rand[0, 1)$	30	$[-1.28, 1.28]^n$	F_{4Min} *(0,0,…,0)=0*
Generalized Rastrigin	$f_5(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	$[-5.12, 5.12]^n$	F_{5Min} *(0,0,…,0)=0*
Ackley	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^{n}\cos(2\pi x_i)}{n}\right) + 20 + e$	30	$[-32, 32]^n$	F_{6Min} *(0,0,…,0)=0*
Generalized Griewank	$f_7(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$	F_{7Min} *(0,0,…,0)=0*
Generalized Schwefel	$f_8(x) = -\frac{1}{n}\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500, 500]^n$	$F_{8Min}\left(x_i^{optimum}\right) = -418.983$ $x_i^{optimum} = -420.968746$

The results are given in Table 2 and the values of the cost function (which are test functions themselves) are shown in Figures 3–10.

The symbols used in Table 2, $J_{\min}$ and $C_i^{(k)}(d) = \sum_{q=1}^{d} f_i^{(k)}, d = 1, 2, \ldots N_i T_{sim}$, are the smallest value of the performance index (the cost function) and the simulation time, respectively. In Table 2, m is the optimal interval and *iteration* is the iteration in which the optimal interval is obtained.

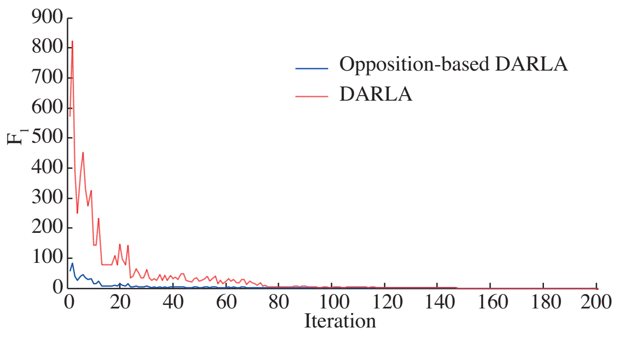


Figure 3. Performance comparison between DARLA and the opposition-based DARLA for F_1.

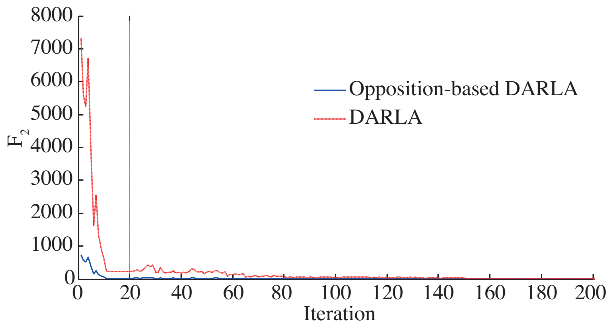


Figure 4. Performance comparison between DARLA and the opposition-based DARLA for F_2.

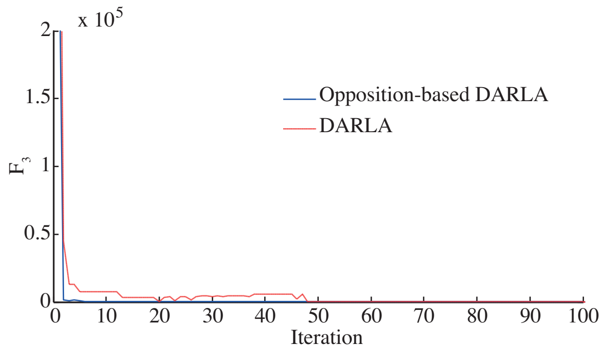


Figure 5. Performance comparison between DARLA and the opposition-based DARLA for F_3.

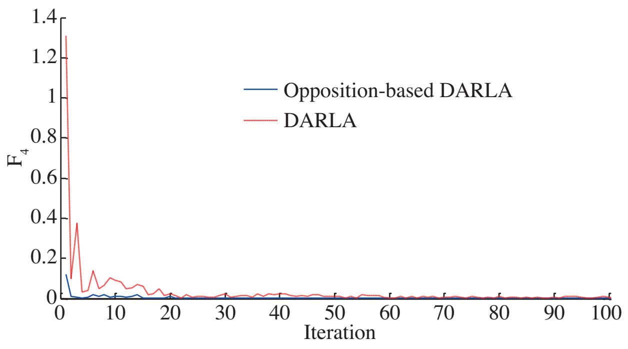


Figure 6. Performance comparison between DARLA and the opposition-based DARLA for F_4.

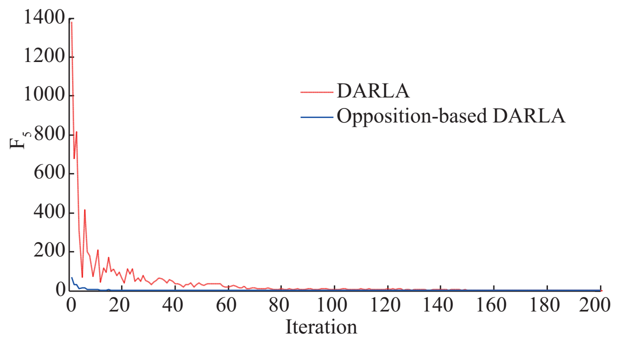


Figure 7. Performance comparison between DARLA and the opposition-based DARLA for F_5.

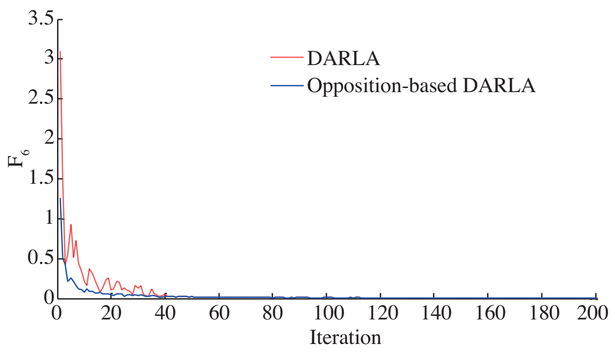


Figure 8. Performance comparison between DARLA and the opposition-based DARLA for F_6.

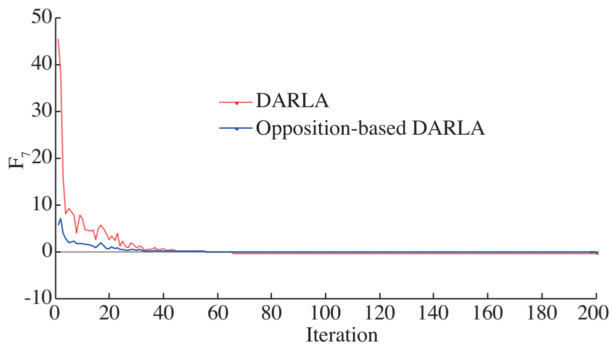


Figure 9. Performance comparison between DARLA and the opposition-based DARLA for F_7.

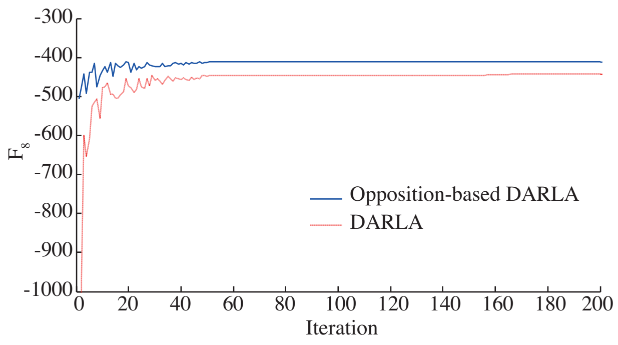


Figure 10. Performance comparison between DARLA and the opposition-based DARLA for F_8.

Table 2. A comparison between DARLA and the opposition-based DARLA.

Test function	DARLA			Opposition-based DARLA		
	$J_{\min}$	*iteration*	T_{sim}	$J_{\min}$	*iteration*	T_{sim}
F_1	0	133	0.09446	0	98	0.03530
F_2	0	162	0.09536	0	114	0.04118
F_3	50.81	68	0.05651	0.01785	41	0.02228
F_4	0.5186	199	0.05951	0	45	0.03471
F_5	0.3162	200	0.057202	0.00386	164	0.01612
F_6	0	182	0.058602	0	113	0.04166
F_7	–0.00760	43	0.067898	0.00034	27	0.03958
F_8	417.4	57	0.059541	416	42	0.5621

In Table 3, the results of test functions for DARLA and its opposite-based version are summarized. For each function, the best, median, worst, mean, and standard deviation of 25 runs are presented.

All of the results have been averaged over 25 runs per test function. The best, mean, and worst in Table 3 indicate the best solution, average of the solutions, and the worst solution among the solutions of 25 runs, respectively, and 'Std dev' stands for the standard deviation. These results show the robustness of the proposed algorithm.

Table 3. DARLA and the opposition-based DARLA.

Test function		F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8
DARLA	Best	0	9.8×10^{-9}	51.73	0.5327	0. 3958	2.6×10^{-12}	0.3630	–419.24
	Mean	0	1.6×10^{-7}	65.77	0.7198	0.6290	3.5×10^{-9}	0.9813	–432.68
	Worst	0	3.27×10^{-7}	89.65	0.95123	0.9229	9.2×10^{-9}	1.0527	–446.239
	Std dev	0	1.8×10^{-7}	38.78	3.5×10^{-2}	0.2893	6.4×10^{-9}	0.271	20.234
Opposition-based DARLA	Best	0	0	0.02472	3.8×10^{-18}	4.6×10^{-3}	0	4.2×10^{-4}	–418.13
	Mean	0	0	2.5×10^{-2}	6.2×10^{-14}	4.1×10^{-2}	1.1×10^{-16}	6.1×10^{-4}	–423.309
	Worst	0	0	0.03356	1.5×10^{-14}	4.3×10^{-2}	2.7×10^{-16}	7.9×10^{-4}	–428.132
	Std dev	0	0	1.1×10^{-2}	3.4×10^{-14}	2.1×10^{-2}	3.8×10^{-16}	3.2×10^{-5}	–4.3459

5.2. AVR system

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 reliability of the power system. Due to the simplicity and low cost of the analog PID, it is still used as the AVR controller. However, the parameters of PID controller are not tuned easily [11,12]. In this paper, the AVR system under study has been modeled based on the IEEE standard in [11] and [12]. The AVR system model controlled by the PID controller can be expressed by Figure 11.

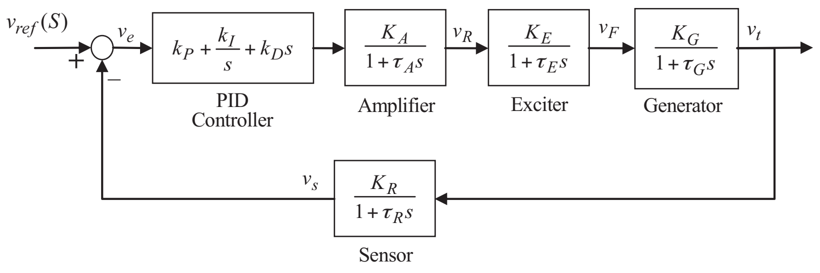


Figure 11. Closed-loop block diagram of the AVR system.

Here, v_s is the output voltage of the sensor, v_e is the error voltage between v_s and the reference input voltage$v_{ref}\,(S)$, v_R is an amplified voltage made by the amplifier, v_F is a output voltage of the exciter model, and $v_t$is the output voltage of the generator [11,13]. The transfer functions of the AVR components can be represented as follows: 1) PID controller model, 2) amplifier model, 3) exciter model, 4) generator model, and 5) sensor model. Their transfer functions are described in Figure 11 [11].

In this plant, K_i is a gain and τ_i is a time constant. The AVR parameters and some other symbols used in this plant are mentioned in Table 4 [11,12].

Table 4. The AVR system parameters [11,12].

Description	Parameter	Value
Amplifier gain	K_A	10
Amplifier time constant	τ_A	0.1
Exciter gain	K_E	1
Exciter time constant	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

Selection of the cost function is very important. While the suitable performance of the controller is dependent on the quality of tracking the reference input (i.e. the unit step), the cost function must be a combination of the important tracking parameters. Here, an effective cost function is suggested for designing the PID controller [1,3,4]:

$$J = G_e \int_0^\tau te^2\,(t)\,dt + G_u \int_0^\tau u_c^2\,(t)\,dt + G_M M_p + G_s M_p + G_s E_{ss} + G_d \sup_t \left| \frac{de\,(t)}{dt} \right|, \tag{11}$$

where $x_i \in [a_i, b_i]$ is the total simulation time and must be sufficiently large, i.e. for this study, $x_i \in [a_i, b_i] =$ 20 s; $e(t)$ 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; and the coefficients denoted as G are the weights of the cost function elements [1–4,6]. Their values are: $G_e = 10$, $G_u = 1$, $G_M = 10$, $G_s = 10$, and $G_d = 5$ [1,3,4]. The normal ranges of the 3 controller parameters that are k_P, k_I, and k_D, considered in [11,12,14], are available and are taken as [0,1.5], [0,1], and [0,1], respectively. However, we want to show our algorithm's robustness and convergence rate, so these intervals are taken as [0,10].

Each term in Eq. (11) has a special effect on the cost function; $\int_0^\tau te^2(t)\,dt$ is the integral of time multiplied by the squared error performance criterion and makes the tracking error diminish in the shortest time [14], $\int_0^\tau u_c^2(t)\,dt$ limits the control energy, M_p and E_{ss} decrease the overshoot and steady-state error, respectively, and $\sup\limits_t \left|\frac{de(t)}{dt}\right|$ makes the response track the reference input smoothly [1,3,4].

Figure 12 shows the original step response of the AVR system without a controller. In this case study, $M_P\% = 50.51$, $E_{ss}\% = 9.09$, $T_r = 0.2843$ s, and $T_s = 6.153$. These parameters are the overshoot percentage, steady-state error percentage, rise time, and settling time, respectively.

The PID controller parameters are calculated by the Ziegler–Nichols method [11,12,14]:

$$k_P = 1.02, k_I = 0.31, k_D = 0.1847.$$

The proposed algorithm (opposition-based DARLA) and the DARLA method are run and their results are given in Table 5. The AVR system with these controller gains is then simulated and the results are shown in Figure 13.

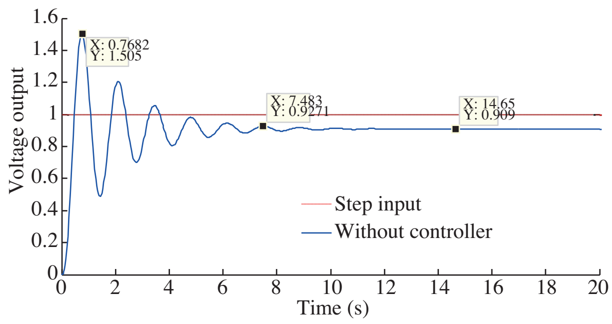


Figure 12. Step response of the AVR system without a controller.

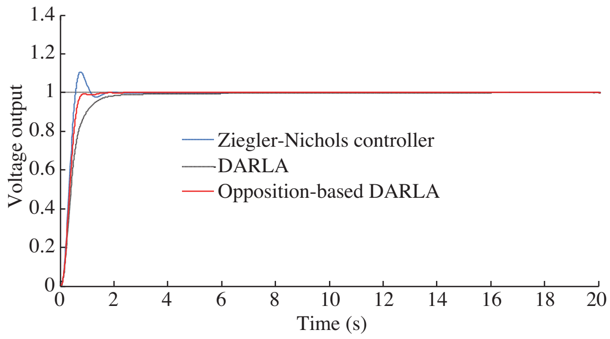


Figure 13. The step response of the AVR system corresponding to the Ziegler–Nichols, DARLA, and opposition-based DARLA controllers.

Table 5. A comparison between the opposition-based DARLA and DARLA.

Iteration	DARLA							Opposition-based DARLA						
	k_P	k_I	k_D	$M_P\%$	$E_{ss}\%$	$J_{\min}$	T_{sim}	k_P	k_I	k_D	$M_P\%$	$E_{ss}\%$	$J_{\min}$	T_{sim}
5	0.605	0.621	0.200	19.5	0.056	31.985	1.745	2.5602	1.0204	0.910	7.6	0	24.5345	0.9273
20	0.900	0.460	0.186	2.74	0	19.5194	4.0815	1.0607	0.232	0.4148	1.24	0	16.8599	3.9862
100	0.809	0.172	0.264	0	0	10.452	15.848	0.9076	0.2156	0.3092	0	0	8.4646	9.3122
200	0.834	0.175	0.284	0	0	6.9504	25.8654	0.948	0.256	0.232	0	0	4.8695	18.9883

The symbols used in Table 5 are $J_{\min}$, T_{sim}, $M_P\%$, and $v_eE_{ss}\%$, which represent the minimum value of the performance index (the cost function), simulation time, percentage of the overshoot, and percentage of the steady-state error, respectively.

The results in Table 5 and Figure 13 prove that the proposed algorithm is robust and converges fast, so it can optimize the PID controller parameters quickly and efficiently.

6. Conclusion

In this paper, the performance of a reinforcement learning algorithm is improved using opposite concepts. The proposed method is the opposition-based DARLA. By using opposite concepts, the convergence rate and accuracy can be increased and the method is applicable for many real-time applications. The superior performance, robustness, and efficiency of the proposed method have been proven through extensive simulation results, including the comparison between DARLA and the opposition-based DARLA operation for some benchmark functions and an AVR system. The results show that this approach is robust and suitable for optimizing various control problems, including adaptive control systems with large-scale dimensions.

References

[1] F. Mohseni Pour, A.A. Gharaveisi, A. Afroomand, S.M.A. Mohammadi "Optimizing a fuzzy logic controller for a photovoltaic grid independent system", 1st Annual Clean Energy Conference on International Center for Science, High Technology & Environmental Sciences pp. 237–241 2010.

[2] M. Kashki, A. Gharaveisi, F. Kharaman, "Application of CDCARLA technique in designing Takagi-Sugeno fuzzy logic power system stabilizer (PSS)", IEEE International Conference on Power and Energy, pp. 280–285, 2006

[3] G. Heydari, A.A. Gharaveisi, M. Rashidinejad, "Optimized PI controller design in motor speed control by composition reinforcement learning automata approach" 1st Annual Clean Energy Conference on International Center for Science, High Technology & Environmental Sciences, pp. 69–76, 2010.

[4] G. Heydari, A.A. Gharaveisi, S.M.R Rafie, "Optimized PID controller design in voltage control of boost regulator by composition reinforcement learning automata approach", 1st Annual Clean Energy Conference on International Center for Science, High Technology & Environmental Sciences, pp. 204–211, 2010.

[5] T. Fukuda, Y. Hasegawa, K. Shimojima, F. Saito, "Reinforcement learning method for generating fuzzy controller", IEEE International Conference on Evolutionary Computation, Vol. 1, pp. 273–278, 1995.

[6] HR. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence", Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation, Vol. 1, pp 695–701, 2005.

[7] HR. Tizhoosh, "Opposition-based reinforcement learning", Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol. 10 pp. 579–586 2006.

[8] HR. Tizhoosh, "Reinforcement learning based on actions and opposite actions", International Artificial Intelligence and Machine Learning Conference pp. 94–98 2005

[9] H.R. Tizhoosh, M. Ventresca, Oppositional Concepts in Computational Intelligence, New York, Springer, 2008.

[10] X Yao, Y Liu, G. Lin, "Evolutionary programming made faster", IEEE Transactions on Evolutionary Computation, Vol. 3, pp. 82–102, 1999.

[11] Z.L. Gaing, "A particle swarm optimization approach for optimum design of PID controller in AVR system," IEEE Transactions on Energy Conversion, Vol. 19, pp. 384–394 ,2004.

[12] H. Yoshida, K. Kawata, Y. Fukuyama, Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage security assessment," IEEE Transactions on Power Systems, Vol. 15, pp. 1232–1239, 2000.

[13] F. Naderi, A.A. Gharaveisi, M. Rasidinejad, "Optimal design of type_1 TSK fuzzy controller using GRLA for AVR system", Large Engineering Systems Conference on Power Engineering, pp. 106–111, 2007.

[14] K.H. Ang, G. Chong, Y. Li, "PID control system analysis, design, and technology", IEEE Transactions on Control System Technology, Vol. 13, pp. 559–576, 2005.