Evolutionary Fuzzy Logic Controller Schemes Suitable for the Simple Automatic Design

Kidchop Waiyasusri
Department of Computer Engineering,
Chulalongkorn University
Bangkok, Thailand
47718076@student.netserv.chula.ac.th

Prabhas Chongstitvatana
Department of Computer Engineering,
Chulalongkorn University
Bangkok, Thailand
prabhas@chula.ac.th

***Abstract*— This paper proposes three styles of the Evolutionary Fuzzy Logic Controller (EFLC) scheme, which are adapted from the Genetic-based Fuzzy Logic Controller (GFLC) proposed in [1]. The main idea of EFLC is the simple 4th order polynomial equation usages for input-output relationship calculations instead of fixed-membership function calculation style in the original GFLC. Although the usage of polynomial equations makes the proposed EFLC to have more parameters than the GFLC, the EFLC has more flexible adaptation/tuning capability and can be adapted to other applications, such as the fuzzy modeling problem. In addition, the proposed scheme still be suitable for simple automatic parameter tuning techniques, such as (1+1)-ES, and have low computation burden. The performance of the proposed scheme comparing with the GFLC is shown by two different experiments: the automatic controllers tuning and the fuzzy modeling of three-dimensional surfaces. The results show that the EFLC can be working well as the controller, and also can be applied to the fuzzy modeling problem.**

***Keywords*—Fuzzy, Controller, FLC, Evolutionary, ES, Automatic Design**

I. INTRODUCTION

Fuzzy logic Controller (FLC) is an application using the Fuzzy Logic System (FLS) to model a controller from the human/expert knowledge. The usage of fuzzy linguistic variables and fuzzy rule-based approximate reasoning makes FLC dealing well with uncertainties and nonlinearity of the controlled system plant. However, to construct FLC, the good knowledge, which leading to the appropriated membership functions and fuzzy rules, is required. If there is not adequate knowledge, the FLC design may be based on the trial-and-error experiment making the FLC design much harder and the good FLC may not be gotten.

The FLC design problem can be looking as the parameter optimization problem; thus, to overcome this problem, there are many approaches that have been developed to generate FLC automatically using Evolutionary Algorithms (EA), which is one of the powerful intelligent search/optimization technique that suited to complex problem [2]-[9]. Furthermore, there are some works trying to simplify the automatic design process by minimizing the number of FLC's parameters [1], [2]. One of those works is the Genetic-based Fuzzy Logic Controller (GFLC) scheme, which is designed to control the speed of motor and has just three parameters to be tuned [1].

The main processes of the GFLC are to transform two inputs of the controller, error and its derivative (change in error), to be a vector in polar coordinate form and then use the fixed-membership function style to calculate the controller's output. The detail of GFLC is discussed later in Section II.

Although the GFLC scheme can reduce the number of parameter to be just three, the fixed-membership function style of the controller may be not suitable for some types of the controlled system plant. Thus, in this paper, three styles of the evolutionary fuzzy logic controller (EFLC) scheme, adapted from the GFLC scheme, are presented.

The main objective of the EFLC scheme is to construct the FLC system that is suitable for simple automatic design, such as Evolutionary Strategies (ES) [10], and can be applied to wider range of problem, not just only for the controller design problem. The EFLC scheme is described in Section III, and the experiment results comparing the performance of EFLC and GFLC are shown in Section IV.

II. GENETIC-BASED FUZZY LOGIC CONTROLLER SCHEME

Giving that $e(k)$, $de(k)$ are the error signals and its derivative respectively , $U(k)$ is the stabilizing output and $u(k) = U(k) + U(k-1)$ is the controller output at time step k , the GFLC scheme can be described as following calculation steps:

Step 1: Compute the scaled change of error, $de'(k)$, using

$$de'(k) = de(k).F_a \tag{1}$$

Step 2: Calculate $R(k)$ and $\theta(k)$ using

$$R(k) = \sqrt{e(k)^2 + de'(k)^2} \tag{2}$$

$$\theta(k) = \tan^{-1}(de'(k) / e(k)) \tag{3}$$

Step 3: Compute the value of membership function $T_N(\theta)$ and $T_P(\theta)$ as follows (see Fig. 1(a)),

$$T_N(\theta)=\begin{cases}1 & 0\le\theta\le\pi/2\\ 2(\pi-\theta)/\pi & \pi/2<\theta\le\pi\\ 0 & \pi<\theta\le 3\pi/2\\ 2(\theta-3\pi/2)/\pi & 3\pi/2<\theta\le 2\pi\end{cases} \quad (4)$$

$$T_P(\theta)=\begin{cases}0 & 0\le\theta\le\pi/2\\ 2(\theta-\pi/2)/\pi & \pi/2<\theta\le\pi\\ 1 & \pi<\theta\le 3\pi/2\\ 2(2\pi-\theta)/\pi & 3\pi/2<\theta\le 2\pi\end{cases} \quad (5)$$

Step 4: Determine the gain value $G_c(k)$ (see Fig. 1(b)),

$$G_c(k)=\begin{cases}R(k)/D_r & \forall R(k)\le D_r\\ 1.0 & \forall R(k)>D_r\end{cases} \quad (6)$$

Step 5: Compute the stabilizing output $U(k)$ using

$$U(k)=G_c(k)[T_N(\theta)-T_P(\theta)]U_{\max} \quad (7)$$

It can be seen from these calculation steps that the main tuning parameters of the GFLC are F_a, D_r and $U_{\max}$. The function of these parameters may be described as follow:

- F_a is used to scale $de(k)$ input signal to the appropriated level, which is related to represent the relationship between two controller inputs.
- D_r is used to determine the controller output signal slope related to the magnitude ($R(k)$) of the controller inputs.
- $U_{\max}$ is using as the maximum value of the controller output signal.

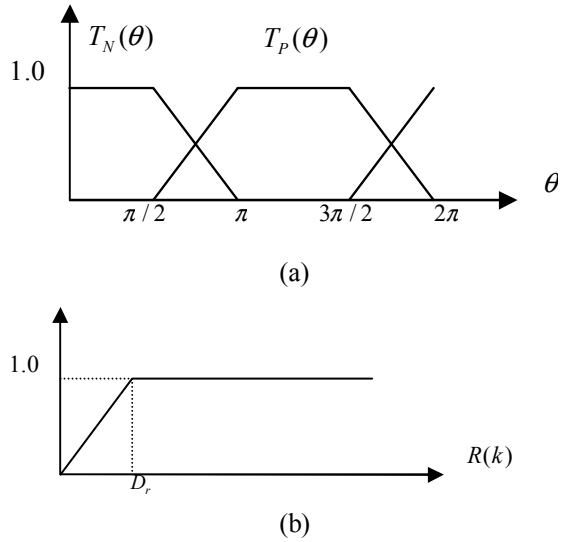

Figure 2. Membership functions for (a) θ (b) R.

III. EVOLUTIONARY FUZZY LOGIC CONTROLLER SCHEME

From the GFLC scheme detail in the previous section, we can see that the rule part of the GFLC is hidden in $[T_N(\theta)-T_P(\theta)]$ term and $G_c(k)$ function. The $[T_N(\theta)-T_P(\theta)]$ term acts mainly as the direction rule, while $G_c(k)$ function acts as the magnitude rule. However, the three tuning parameters of the controller are quite not involved in those rule terms. The only parameter that has effect with the rule terms is D_r that determined the slope of controller output, while F_a and $U_{\max}$ just function as the normal scaling parameters. This let the GFLC acting like the fixed-rule FLC, as same as the FLC proposed in [2], leading to the low flexible controller system and obviously can not be applied to other applications.

To overcome this problem, we propose the Evolutionary Fuzzy Logic Controller (EFLC) that modifies the GFLC scheme by replacing the calculation step 3 and 5 as follows:

Step 3N1: Calculate $T(\theta)$ using

$$T(\theta)=c_0+c_1\theta+c_2\theta^2+c_3\theta^3 \quad (8)$$

when $T(\theta)\in[-1,1]$

Step 5N1: Compute the stabilizing output $U(k)$ using

$$U(k)=G_c(k)T(\theta)U_{\max} \quad (9)$$

These replacements make the EFLC to have four more parameters to be tuned: c_0, c_1, c_2 and c_3, or totally seven parameters. The main objective of using 4th order polynomial is to get more flexible controller structure with not too much parameters. This EFLC style will be called "EFLC-1".

Moreover, if we want to give more flexibility to the controller structure, we can change the $G_c(k)$ in step 4 to be polynomial equation as shown in equation (10) and change the controller output calculation in step 5 to be (11). We will call this system "EFLC-2". The EFLC-2 has just two more parameters than the EFLC-1 because of the reduction of unused D_r and $U_{\max}$.

Step 4N2: Calculate $G_c(k)$ using

$$G_c(t)=cg_0+cg_1R+cg_2R^2+cg_3R^3 \quad (10)$$

Step 5N2: Compute the stabilizing output $U(k)$ using

$$U(k)=G_c(k)T(\theta) \quad (11)$$

Anyway, we can also adapted the EFLC-2 by adding the limitation checking step for the gain value $G_c(k)$ after step 4N2 as shown below (step 4-1) to give an output-limitation ability to the control scheme. We will call this adapted EFLC-2 as "EFLC-3". This scheme has totally 10 parameters for tuning.

Step 4-1: Limit $G_c(k)$ using

$$G_c(k) = \begin{cases} U\max & Gc(k) > U\max \\ 0 & Gc(k) < 0 \end{cases} \quad (12)$$

EFLC-1, EFLC-2 and EFLC-3 scheme has 7, 9 and 10 tuning parameters respectively. These numbers of parameters are more than the three parameters in the original GFLC, but they still less than 15 parameters of the fixed-rule FLC proposed in [2]. In addition, the experimental results in the next section will show that the proposed EFLC schemes still can be easy tuning by ES and give the better results than the original GFLC.

IV. EXPERIMENTS AND RESULTS

A. Controller Design Experiment

For this controller design experiment, the closed-loop step responses of three process plants, which are used as the test plants in [2], [3], are also be observed in this paper.

Plant A: $$G_A(s) = \frac{2}{s(s+1.4)+2} \quad (13)$$

Plant B: $$G_B(s) = \frac{2}{(s+1)(s+2)} \quad (14)$$

Plant C: $$G_C(s) = \frac{1}{s(1+0.1s)} \quad (15)$$

Firstly, we apply (1+1)-ES technique with maximum of 100 generations to automatic design/tuning the controller parameters. The Integral-of-Time-multiplied Absolute-Error (ITAE) is adopted to be the fitness function as following equation:

$$fitness = 100\int_0^T t\left|error(t)\right|dt \quad (16)$$

where dt is the sampling period
T is the total running time

The results of 5 runs for each FLC (GFLC, EFLC-1, EFLC-2 and EFLC-3) tuning are shown in Table I. The simulation are done in MATLAB with 0.01s sampling period and 10s total simulation time.

From Table I, we can see that all of the proposed EFLCs can do much better than the GFLC for Plant A and B. For Plant C results, the original GFLC has the best average fitness function value with low variation and EFLC-2 is the worse one, but EFLC-1 and EFLC-3 have the better results for the best fitness value results.

The Plant C results are very interesting because we can see that EFLC-1, which has the closest structure to the GFLC, has the good results and quite close to the GFLC's results, while EFLC-2 and EFLC-3, which have more flexible structure, can not do quite well. These results may lead to the conclusion that the GFLC has already had the appropriated structure to control Plant C, which also making EFLC-1 can do well for Plant C.

TABLE I. THE RESULTS OF FLCS TUNING BY (1+1)-ES

Plant	Fuzzy System	Best Fitness	Avg. Fitness	Worst Fitness.
G_A	GFLC	109.84	136.6	232.94
	EFLC-1	23.56	84.67	215.36
	EFLC-2	13.62	39.23	87.77
	EFLC-3	9.87	19.11	49.54
G_B	GFLC	37.98	49.84	67.91
	EFLC-1	6.23	18.44	31.52
	EFLC-2	6.83	24.95	51.06
	EFLC-3	9.03	24.24	65.22
G_C	GFLC	4.84	5.31	6.94
	EFLC-1	3.15	6.42	13.8
	EFLC-2	9.49	49.72	115.08
	EFLC-3	2.99	31.01	103.76

Next, we use $(\mu/\rho+\lambda)$-ES with only one standard deviation (c) to tune the parameters of FLCs. The setting of $(\mu/\rho+\lambda)$-ES are: $\mu = \rho = \lambda = 2$, intermediate recombination and the maximum generations is 100. The results of 3 runs for each FLCs are shown in Table II.

TABLE II. THE RESULTS OF FLCS TUNNING BY ($\mu/\rho+\lambda$)-ES

Plant	Fuzzy System	RUN1 Fitness	RUN2 Fitness	RUN3 Fitness	Avg. Fitness
G_A	GFLC	113.93	113.28	117.48	114.89
	EFLC-1	19.00	29.69	32.24	26.97
	EFLC-2	9.35	5.52	12.48	9.12
	EFLC-3	25.81	15.78	9.49	17.03
G_B	GFLC	48.49	44.56	48.08	47.04
	EFLC-1	25.85	28.69	39.53	31.36
	EFLC-2	5.43	10.01	14.84	10.09
	EFLC-3	19.00	9.56	21.73	16.76
G_C	GFLC	4.85	4.84	4.86	4.85
	EFLC-1	0.98	4.86	2.40	2.75
	EFLC-2	20.74	40.26	7.14	22.71
	EFLC-3	15.1	17.34	20.58	17.67

The results of using $(\mu/\rho+\lambda)$-ES shows that the GFLC has very low variation from run to run, while the proposed EFLCs have a lot more variation, especially for EFLC-2 and EFLC-3. This result can be easily explained by the more parameters and flexibility of the EFLC-2 and EFLC-3 structure. In addition, the results in Table II still lead to the same conclusion from Table I that all of the proposed EFLCs can do much better

than the GFLC for Plant A and B. However, the results of Plant C have one main difference: EFLC-1 is the best one for Plant C, which may be the effect from the more complex searching algorithm of $(\mu/\rho+\lambda)$-ES. This difference can lead to deeper explanation for Plant C results: the GFLC has already had the appropriated structure of gain value function $G_c(k)$ to control Plant C, and the EFLC-1 that has more flexible in $T(\theta)$ part can be tuned to get the better control result.

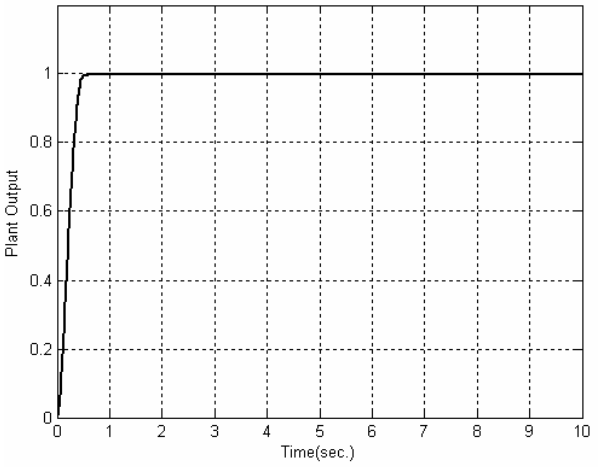

Figure 2. Example step response for Plant C with EFLC-1 controller

B. Fuzzy Modeling Problem Experiment

Because of the more flexible structure, we believe that the proposed EFLCs can be applied to other problem, more over than a controller design problem. In this experiment, we use the proposed EFLCs and GFLC to model following three-dimensional surfaces [4] (see Fig. 3):

$$fx_1(x,y) = x^2 + y^2; \quad x,y \in [-5,5], fx_1 \in [0,50] \qquad (17)$$

$$fx_2(x,y) = \frac{x-xy}{x-2.xy+y}; \quad x,y \in [0,1], fx_2 \in [0,10] \qquad (18)$$

The inputs of the tested FLCs are changed to be x and y instead of $e(k)$ and $de(k)$ respectively. The output of FLCs is changed from $U(k)$ to be $f'(x,y)$. Moreover, we will use $(\mu/\rho+\lambda)$-ES with the setting: $\mu=\rho=2, \lambda=10$, intermediate recombination and the maximum generations is 100, as the tuning algorithm. The fitness function for tuning is the Mean Square Error (MSE) as shown in (19).

$$fitness = \frac{\sum_{i=1}^{N}(fx(x_i,y_i) - f'(x_i,y_i))^2}{N} \qquad (19)$$

where i is the index of input data set
N is the number of training point

TABLE III. THE RESULTS OF SURFACE MODELING PROBLEM

Func.	Fuzzy System	RUN1 Fitness	RUN2 Fitness	RUN3 Fitness	Avg. Fitness
fx1	GFLC	40.84	40.85	40.84	40.84
	EFLC-1	31.34	24.17	24.09	26.53
	EFLC-2	2.77	6.60	1.70	3.69
	EFLC-3	26.36	33.32	56.50	38.73
fx2	GFLC	13.07	13.07	13.07	13.07
	EFLC-1	7.11	13.07	11.86	10.68
	EFLC-2	10.54	7.62	7.19	8.45
	EFLC-3	12.34	12.05	8.18	10.86

The experimental results of 3 runs for each FLCs are shown in Table III. It is obvious that the GFLC has the worse result because of its fixed-rule style. In addition, from Table III, it can be seen that EFLC-2 has the best result, which may be the effect of its flexible scheme without the output limitation. The examples of graphical results are shown in Fig. 4. From Fig. 4, we can observe that EFLC-2 can do well in modeling $fx1$ but not for $fx2$. This result happens because equation of $fx1$ is equivalent to the magnitude rule part ($G_c(k)$) of EFLC-2, while $fx2$ may be too complex to the proposed EFLC structures.

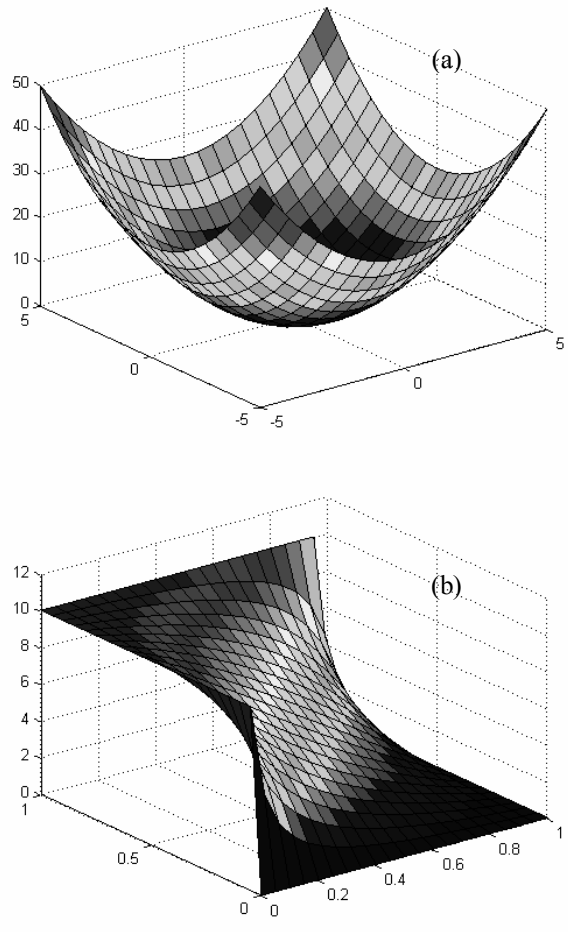

Figure 3. Graphical representations of (a) $fx1$ (b) $fx2$

V. CONCLUSION

This paper presents three styles of the EFLC scheme adapted from the GFLC. All of the proposed EFLC can be done well in the automatic controller design problem with just an easy ES technique as the tuning algorithm. Moreover, the proposed schemes can also be used in the fuzzy modeling problem quite well as shown in the experiments.

VI. FUTURE WORK

The structure of EFLC can be extended to deal with the three-input problem. It may be done by transform the inputs to be in the sphere coordinate form and apply the two-input EFLC structure to calculate the direction rule path. Moreover, it would be interesting to apply the EFLC to some real world applications.

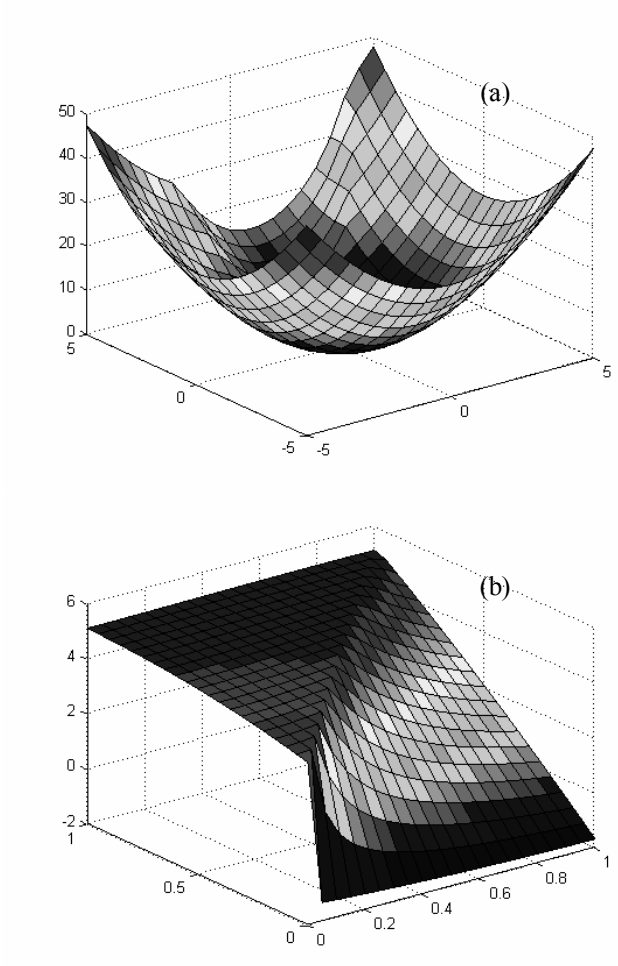

Figure 4. Example of the graphical results getting from EFLC-2 for (a) $fx1$ (b) $fx2$

APPENDIX

Evolutionary Strategies (ES) [10] are the techniques of Evolutionary Algorithms (EA) that were developed by Rechenberg and Schwefel to solve real-valued parameter optimization problems. The first and simplest ES algorithm, the so-called (1+1)-ES, was introduced with just two individuals (populations) and only one mutation operator. Then, it was developed to have more searching/optimization ability by adding more parents and offspring, recombination stage and self-adaptation for mutation operator. Two ES techniques referred in this paper, (1+1)-ES and $(\mu/\rho+\lambda)$-ES with one self-adaptive strategy parameter (c), are introduced to give more understanding.

A. *(1+1)-ES*

In (1+1)-ES, there are just two individuals per generation, one parent and one offspring. For each generation, the parent individual is mutated to generate one offspring with the mutation/strategy parameter, a standard deviation (c). Giving that **parent** $=< x_1, x_2, ..., x_n >$, the mutation steps can be show as follows:

Step 1: Calculate mutation step $\mathbf{z}$,

$$\mathbf{z} = \sigma . < N_1(0,1), N_2(0,1), ..., N_n(0,1) > \qquad \text{(A-1)}$$

where $N_i(0,1)$ are independent random samples from the standard normal distribution

Step 2: Obtain offspring using,

$$\mathbf{offspring} =< y_1, y_2, ..., y_n >= \mathbf{parent} + \mathbf{z} \qquad \text{(A-2)}$$

After mutation, we then select the best individual (best fitness value) to be parent in next generation. Moreover, the strategy parameter (c) is also evolved by Rechenberg's 1/5-success rule:

$$\sigma = \begin{cases} \sigma / a & if\ P_S > 1/5 \\ \sigma a & if\ P_S < 1/5 \\ \sigma & if\ P_S = 1/5 \end{cases} \qquad \text{(A-3)}$$

where a is a constant : $0.85 < a < 1$

P_S is the success probability value:

$$P_S = \frac{G_S}{G} \qquad \text{(A-4)}$$

where G_S is the number of generations that offspring is better than parent

G is the number of current generation

These steps are repeated until the solution is obtained or the termination condition is true.

B. *$(\mu/\rho+\lambda)$-ES with one self-adaptive c*

This kind of ES has μ parents and λ offspring for each

generation. One strategy parameter c is added to individual parameters to be evolved (mutated) together instead of using 1/5-success rule; thus, an individual is in the form:

$$\textbf{individual} =< x_1, x_2, ..., x_n, \sigma > \qquad \text{(A-5)}$$

Before mutation, ρ parent individuals are uniform-randomly chosen from μ parents to be recombined. One type of recombination is the intermediate recombination, which can be shown as follow:

$$\mathbf{r} =< \frac{\sum_{i=1}^{\rho} x_{i1}}{\rho}, \frac{\sum_{i=1}^{\rho} x_{i2}}{\rho}, ..., \frac{\sum_{i=1}^{\rho} x_{in}}{\rho} > \qquad \text{(A-6)}$$

where $\mathbf{r}$ is a recombined individual

x_{ij} are parameters of i^{th} chosen individual

After that, an obtained recombined individual is mutated in three steps:

Step 1: Mutate c first,

$$c' = c \exp(\tau N(0,1)) \qquad \text{(A-7)}$$

where τ is the learning parameter: $\tau \propto 1/\sqrt{n}$

Step 2: Calculate mutation step $\mathbf{z}$,

$$\mathbf{z} = \sigma' < N_1(0,1), N_2(0,1), ..., N_n(0,1) > \qquad \text{(A-8)}$$

Step 3: Obtain offspring using,

$$\textbf{offspring} = \mathbf{r} + \mathbf{z} \qquad \text{(A-9)}$$

The steps of recombination and mutation are repeated to get λ offspring individuals, and then select the best μ individuals from μ parents and λ offspring to be the parent individual. The described steps are repeated until the solution is obtained or the termination condition is true.

REFERENCES

[1] M.N. Uddin, M.A. Abido, M.A. Rahman, "Real-Time Performance Evaluation of A Genetic-Algorithm-Based Fuzzy Logic Controller for Motor Drives," IEEE Trans. Industrial Applications, vol. 41, NO. 1, pp. 246-252, 2005.

[2] France Cheong, Richard Lai, "On Simplifying the Automatic Design of a Fuzzy Logic Controller," Proc. NAFIPS, pp. 481-487, 2002.

[3] T.Y. Huang, Y.Y. Chen, "Generation of a Fuzzy Logic Controller Using Evolutionary Strategies," IFSA World Congress and 20th NAFIPS Int. Conf., pp. 269-274, Jul. 2001.

[4] Oscar Cordon, Francisco Herrera, "A Two-Stage Evolutionary Process for Designing TSK Fuzzy Rule-Based Systems," IEEE Trans. Systems, Man, And Cybernatics-Part B: Cybernatics, vol. 29, NO. 6, pp. 703-715, Dec. 1999.

[5] Frank Hoffman, "Evolutionary Algorithms for Fuzzy Control System Design," Proc. of THE IEEE, vol. 89, NO. 9, pp. 1318-1333, Sept. 2001.

[6] W.R. Hwang, W.E. Thompson, "Design of Intelligent Fuzzy Logic Controllers Using Genetic Algorithms," IEEE World Congress on Computational Intelligence, Proc. Fuzzy Systems, vol. 2, pp.1383-1388, Jun. 1994.

[7] Y.C. Chiou, L.W. Lan, "Genetic Fuzzy Logic Controllers," IEEE Int. Conf. Transportation Systems, pp. 200-205, Sept. 2002.

[8] K.B. Sim, K.S. Byun, D.W. Lee, "Design of Fuzzy Controller Using Schema Coevolutionary Algorithm," IEEE Trans. Fuzzy Systems, vol. 12, NO. 4, pp. 565-568, Aug. 2004.

[9] K. Shimojima, N.Kubota, T. Fukuda, "RBF Fuzzy Controller with Virus-Evolutionary Genetic Algorithm," IEEE Conf. Neural Networks, vol. 2, pp. 1040-1043, Jun. 1996

[10] H.G. Beyer, H.P. Schwefel, "Evolutionary Strategies: A comprehensive introduction," Natural Computing 1, pp. 3-52, 2002.