Modulus Genetic Algorithm and its Application to Fuzzy System Optimization

Sinn-Cheng Lin

Department of Educational Media and Library Sciences,
Tamkang University, Taipei, Taiwan, R.O.C.

ABSTRACT

The conventional genetic algorithm encodes the searched parameters as binary strings. After applying the basic genetic operators such as reproduction, crossover and mutation, a decoding procedure is used to convert the binary strings to the original parameter space. As the result, such an encoding/decoding procedure leads to considerable numeric errors. This paper proposes a new algorithm called modulus genetic algorithm (MGA) that uses the modulus operation to resolve this problem. In the modulus genetic algorithm, the encoding/decoding procedure is not necessary. It has the following advantages: 1) the evolution can be speeded up; 2) the numeric truncation error can be avoided; 3) the precision of solution can be increased.

The proposed MGA is applied to resolve the key problem of fuzzy inference systems – rule acquisition. The fuzzy system with MGA as learning mechanism forms an "intelligent fuzzy system". Based on the proposed approach, the fuzzy rule base can be self-extracted and optimized. Such an intelligent fuzzy system has a general-purpose architecture. It can be applied to many kinds of fields.

INTRODUCTION

Genetic algorithms (GAs) are parallel and global search techniques, which take the concepts from evolution theory and natural genetics. They emulate biological evolution by means of genetic operations such as reproduction, crossover and mutation. Usually, genetic algorithms are used as optimization techniques [1]-[5]. Although there is no necessary and sufficient condition on the functions which are optimizable by genetic algorithms, it has been shown that GAs perform well on multimodal functions (the functions which have multiple local optima). Moreover, various studies have shown that whenever GAs failed to find the optimal solution on a function, other known techniques failed as well [2].

Conventionally, a genetic algorithm works with a set of artificial elements (binary strings, e.g. 10101010), called a population. An individual (string) is referred as a chromosome, and a single bit in the string is called a gene. GA generates a new population (called offspring) by applying the genetic operators to the chromosomes in the old population (called parents). An iteration of genetic operation is referred as a generation. A fitness function, *i.e.* the function to be maximized, is used to evaluate the fitness of an individual. One of the important purposes of GAs is to reserve the better schemata, *i.e.* the patterns of certain genes, so that the offspring may yield higher fitness than their parents. Consequently, the value of fitness function increases from generation to generation. In most of genetic algorithms, mutation is a random-work mechanism to avoid the local optimum trapping problem. As a result, GAs, theoretically, can find the global optimal solution.

The basic disadvantage of the conventional genetic algorithm is that it encodes the searched parameters as binary strings. After applying the basic genetic operators such as reproduction, crossover and mutation, a decoding procedure has to be used to convert the binary strings to the original parameter space. As the result, such an encoding/decoding procedure leads to considerable numeric errors. This paper proposes a new algorithm called modulus genetic algorithm (MGA) that uses the modulus operation to resolve this problem. In the MGA, the encoding/decoding procedure is not necessary. It has the following advantages: 1) the evolution can be speeded up; 2) the numeric truncation error can be avoided; 3) the precision of solution can be increased.

0-7803-5489-3/99/$10.00

The MGA is applied to resolve the key problem of fuzzy logic systems -- rule acquisition [8]. The fuzzy system with MGA as learning mechanism forms an "intelligent fuzzy system". Based on the proposed approach, the fuzzy rule base can be self-extracted and optimized. Such an intelligent fuzzy system has a general-purpose architecture. It can be applied to many kinds of fields, such as fuzzy control, fuzzy image processing, fuzzy decision making, and fuzzy pattern recognition ... etc.

This paper is organized as follows: Section 1 is an introduction. Section 2 describes the detail of the MGA. In Section 3, the MGA is used to build an intelligent fuzzy inference system. Section 4 applied the MGA-based fuzzy inference system to the field of fuzzy control. Conclusions are drawn in Section 5.

THE MODULUS GENETIC ALGORITHM

Reproduction of MGA

The Darwinian "survival of the fittest" is the underlying spirit of reproduction. This operation, actually, is an artificial version of natural selection.

Let F be the fitness function, and F_i denote the value of fitness function associated with the individual string i in the current population. Reproduction is a process in which individual strings in the current population are copied according to their fitness function values F_i. A higher F value indicates a better fit (or larger benefit). To perform reproduction, first, F_i's are calculated. Next, the current individual strings are probabilistically selected and copied into a mating pool according to their fitness value. The arrangement allows the strings with a higher fitness to have a greater probability of contributing a larger amount of offspring in the new population. The easiest way of implementing a reproduction operator is to create a biased roulette wheel. The slot size of it is proportion to the fitness value of each individual in the current population. Let ps_i denote the probability of selection of the individual i, and M be the population size, then an individual string will get selected with the following probability:

$$ps_i = F_i \Big/ \sum_{j=1}^{M} F_j \qquad 1.$$

Crossover of MGA

Crossover provides a mechanism for individual strings to exchange information via a probabilistic process. Once the reproduction operator is applied, the members in the mating pool are allowed to mate with one another. The binary-coded GA takes the following step to accomplish the crossover: First, two parents are randomly selected from the mating pool. Then, a random crossover point is picked up. Finally, exchange the parents' genetic codes (binary digits) following the crossover point. This random process provides a highly efficient method to search the string space to find a better solution.

In MGA, the parameters lie in the original space rather than binary space. Hence, the crossover operation has to be modified to work with parameters themselves rather than their binary codes.

Let $\{a, b\}$ and $\{a', b'\}$ be the parent and offspring parameter pair, respectively. The search space of them is in the range of $[X_{\min}, X_{\max}] \subseteq R$. The crossover of MGA is proposed as follows:

$$\begin{aligned} a' &= (a - a_0 + b_0) \,\mathrm{MOD}\, \Delta + X_{\min} \\ b' &= (b - b_0 + a_0) \,\mathrm{MOD}\, \Delta + X_{\min} \end{aligned} \qquad 2.$$

where MOD means the modulus operator. It is the reason why the proposed approach called "modulus" genetic algorithm. The meaning of other notations in (2) are: $\Delta = X_{\max} - X_{\min}$, and

$$a_0 = a \,\mathrm{MOD}\, \alpha\Delta$$
$$b_0 = b \,\mathrm{MOD}\, \alpha\Delta$$

in which, $\alpha \in [0, 1]$ is called the crossover factor.

Especially, the crossover of a binary-coded GA is a special case of (2) with the following a_0 and b_0:

$$a_0 = a \text{ MOD } 2^c$$

$$b_0 = b \text{ MOD } 2^c$$

where c denotes the bit number of crossover point.

Mutation of MGA

In the genetic algorithm, the mutation operation introduces new genes into the populations such that the problem of trapping in local optimal points may be avoided. The gene of individual is subject to a random change with probability of the pre-assigned mutation rate. In the binary-coded case, a mutation operator changes a bit from 0 to 1 or vice versa. In MGA, mutation is a random work mechanism. It simply replaces a parameter, say a, with an arbitrary value, say a', in the search space $[X_{\min}, X_{\max}]$.

MGA-BASED INTELLIGENT FUZZY SYSTEM DESIGN

Suppose that a fuzzy inference system described by the following rule base [6]:

$$R_j : IF\ x\ is\ X_j(m_j, \sigma_j)\ THEN\ y\ is\ Y_j(\varphi_j) \qquad 3.$$

where $j = 1, 2, \kappa, N$, and N is the number of rules; X_j's and Y_j's are the input and output linguistic labels [10], respectively. Especially, in this paper X_j's are simply assigned as Gaussian-shaped functions, *i.e.*. $\mu_{X_j}(x) = \exp\left(-\left(\frac{x - m_j}{\sigma_j}\right)^2\right)$, and Y_j's are assigned to be fuzzy singletons, *i.e.*, $\mu_{Y_j}(u) = \begin{cases} 1, y = \varphi_j \\ 0, y \neq \varphi_j \end{cases}$.

Suppose that the singleton fuzzification and the weighted average defuzzification methods are applied [8], then the output of (3) is given by:

$$y = \underline{\varphi}^{\mathrm{T}} \underline{\rho}(x) \qquad 4.$$

where $\underline{\varphi} = [\varphi_1, \varphi_2, ..., \varphi_N]^{\mathrm{T}}$ and $\underline{\rho} = [\rho_1, \rho_2, ..., \rho_N]^{\mathrm{T}}$ in which $\rho_j(x) = \frac{\mu_{X_j}(x)}{\sum_{j=1}^{N} \mu_{X_j}(x)}$

Constructing a parameter space to be searched by MGA required transferring the fuzzy rule base (3) to a parameter representation. Clearly, the output of the rule base (3) is uniquely determined by a set of parameters which is unionized by the parameters of IF part and THEN part. Hence, the parameter vector to be learned by genetic algorithm, $\underline{\theta}$, is defined as:

$$\underline{\theta} = [\underline{m}^{\mathrm{T}}\ \underline{\sigma}^{\mathrm{T}}\ \underline{\varphi}^{\mathrm{T}}]^{\mathrm{T}} = [m_1\ m_2 \Lambda\ m_N\ \sigma_1 \sigma_2 \Lambda\ \sigma_N\ \varphi_1 \varphi_2 \Lambda\ \varphi_N]^{\mathrm{T}} \qquad 5.$$

Assume that X_m, X_σ, X_φ are the search space of m_j's, σ_j's, φ_j's, respectively; M is the population size; h is the number of generation. The details of learning procedures of MGA-based fuzzy system are described in the follows.

STEP 1: Initially, set $h = 0$ and randomly generate $3M$ initial parameter vectors,

$$\underline{m}^{(i)}(h) = [m_1^{(i)}(h)\ m_2^{(i)}(h) \Lambda\ m_N^{(i)}(h)]^{\mathrm{T}}$$

$$\underline{\sigma}^{(i)}(h) = [\sigma_1^{(i)}(h) \sigma_2^{(i)}(h) \Lambda\ \sigma_N^{(i)}(h)]^{\mathrm{T}}$$

$$\underline{\varphi}^{(i)}(h) = [\varphi_1^{(i)}(h) \varphi_2^{(i)}(h) \Lambda\ \varphi_N^{(i)}(h)]^{\mathrm{T}}$$

where $m_j^{(i)}(h) \in X_m, \sigma_j^{(i)}(h) \in X_\sigma$ and $\varphi_j^{(i)}(h) \in X_\varphi$ $(i = 1, 2, ..., M, j = 1, 2, ..., N)$.

If the i-th candidate of MGA-based fuzzy inference system is denoted by $\mathrm{FIS}^{(i)}$. Then the fuzzy rule base of $\mathrm{FIS}^{(i)}$ can be created as:

$$\text{FIS}^{(i)}:\begin{cases} R_1^{(i)}: \text{IF } x \text{ is } X_1^{(i)}(m_1^{(i)},\sigma_1^{(i)}) \text{ THEN } y \text{ is } Y_1^{(i)}(\varphi_1^{(i)}) \\ R_2^{(i)}: \text{IF } x \text{ is } X_2^{(i)}(m_2^{(i)},\sigma_2^{(i)}) \text{ THEN } y \text{ is } Y_2^{(i)}(\varphi_2^{(i)}) \\ \text{M} \\ R_N^{(i)}: \text{IF } x \text{ is } X_N^{(i)}(m_N^{(i)},\sigma_N^{(i)}) \text{ THEN } y \text{ is } Y_N^{(i)}(\varphi_N^{(i)}) \end{cases}$$

where $X_j^{(i)}$'s and $Y_j^{(i)}$'s are linguistic labels to be learned, and $m_j^{(i)}$'s, $\sigma_j^{(i)}$'s and $\varphi_j^{(i)}$'s are their parameters.

STEP 2: Construct the parameter vector of the *i*-th individual,

$$\underline{\theta}^{(i)}(h) = [\theta_1^{(i)}(h) \Lambda\ \theta_N^{(i)}(h) \theta_{N+1}^{(i)}(h) \Lambda\ \theta_{2N}^{(i)}(h) \theta_{2N+1}^{(i)}(h) \Lambda\ \theta_{3N}^{(i)}(h)]^T$$
$$= [\underline{m}^{(i)T}(h) \underline{\sigma}^{(i)T}(h) \underline{\varphi}^{(i)T}(h)]^T$$

STEP 3: Establish the population in the generation *h*, $\boldsymbol{P}(h)$,

$$\boldsymbol{P}(h) = \{\theta^{(1)}(h), \theta^{(2)}(h), \text{K}, \theta^{(n)}(h)\}$$

STEP 4: Evaluate the fitness value, $F^{(i)}$, of each individual.

STEP 5: Apply the modulus genetic operators, *i.e.* reproduction of MGA, crossover of MGA and mutation of MGA, to generate a new population $\boldsymbol{P}(h+1)$, which is always known as the offspring of $\boldsymbol{P}(h)$.

STEP 6: Keep the elitist. That is, 1) pick up the best fitted individuals in $\boldsymbol{P}(h)$ and $\boldsymbol{P}(h+1)$; 2) compare their fitness; 3) if the best individual of $\boldsymbol{P}(h)$ has a better fitness value than that of $\boldsymbol{P}(h+1)$, then randomly replace an individual in $\boldsymbol{P}(h+1)$ with the elitist.

STEP 7: Use the parameters to calculate the output *y* of the fuzzy system.

STEP 8: Set $h = h + 1$; go to Step 2 and repeat the procedure until $F \geq F_M$ or $h \geq H$. Where F_M and H denote an acceptable fitness value and a stop generation number, respectively, as specified by the designer.

AN APPLICATION EXAMPLE

The proposed MGA-based intelligent fuzzy system has a general-purpose architecture. It can be applied to many kinds of fields, such as fuzzy control, fuzzy image processing, fuzzy decision making, and fuzzy pattern recognition ... and so on. In this section, an example of MGA-based fuzzy control system is used to demonstrate its practicability. Consider a class of *n*th order nonlinear systems, which is expressed by the following error dynamics [12]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \quad \text{M} \\ \dot{x}_n = f(\underline{x}) + g(\underline{x})u \end{cases} \qquad 6.$$

where $\underline{x} = [x_1\ x_2\ \text{K}\ x_n]^T \in \Re^n$ is the state vector; $u \in \Re$ is the control input; $f(.)$ is an unknown continuous function with known upper bound, *i.e.* $|f| \leq f^U$; $g(.)$ is an unknown positive definite function with known lower bound, *i.e.* $0 < g_L \leq g$. Actually, equation (6) represents a general uncertain nonlinear dynamic system. The chief objectives are:

1) Apply MGA for self-extracting an optimal fuzzy control rule-base, to minimize the following quadratic cost function:

$$J = \frac{1}{2}\int_0^\infty \left(\underline{x}^T(t)\underline{Q}\underline{x}(t) + u^T(t)Ru(t)\right)dt \qquad 7.$$

where $Q \in \Re^{n\times n}$ and $R \in \Re$ are two positive definite weighting matrices.

2) Guarantee the stability of the control system:

$$|x_i| \le \delta_i, i = 1, 2, ..., n \quad 8.$$

To simplify the system design, the fuzzy sliding mode control (FSMC) [12] is adopted as the control scheme. Based on our previous work [14], the control law can be represented as:

$$u = (1-\alpha)u_f + \alpha u_h, \qquad \alpha = \begin{cases} 1, \text{for} |s| \ge \bar{s} \\ 0, \text{for} |s| < \bar{s} \end{cases} \quad 9.$$

where u_f is obtained from the following fuzzy control rule-base:

$$R_j : IF\ s\ is\ S_j(m_j, \sigma_j)\ THEN\ u_f\ is\ U_j(\varphi_j) \quad 10.$$

where $s(x) = \underline{c}^{\mathrm{T}} \underline{x} = \sum_{i=1}^{n} c_i x_i$ is a sliding function and $\underline{c}^{\mathrm{T}} = [c_1\ c_2\ \mathrm{K}\ c_n]^{\mathrm{T}} \in \Re^n$ is the coefficient vector of s. The optimal coefficients of sliding function can be determined by the criterion we proposed in [14].

Notice that the rule base (10) is in the form of (3). Therefore, the approach described in the previous section can be directly applied to find the parameter vector of (10), that is $[m_1\ m_2\ \Lambda\ m_N\ \sigma_1\ \sigma_2\ \Lambda\ \sigma_N\ \varphi_1\ \varphi_2\ \Lambda\ \varphi_N]^{\mathrm{T}}$. To minimize the quadratic cost function (7), the fitness function can be defined as:

$$F = 1 / (J_s + \varepsilon_0)$$

in which $J_s = t_s + \sum_{k=1}^{K} (s^2 + Ru^2)$. Where t_s denotes the reach time of sliding mode; $k = \mathrm{int}(t/\Delta t)$ denotes the iteration instance; Δt is the sampling period; int(.) is the round-off operator; $K = \mathrm{int}(t_{\max}/\Delta t)$ denotes the number of iterations in each run; $t_{\max}$ is the running time in one run.

Moreover, the hitting control law in (9), u_h, is designed to guarantee system stability. If u_h given by [13]:

$$u_h = -sign(s)[g_L^{-1}(f^U + |\bar{c}^T \bar{x}| + \eta)] \quad 11.$$

in which $\bar{c} = [c_1\ c_2\ \mathrm{K}\ c_{n-1}]^{\mathrm{T}}$ and $\bar{x} = [x_1\ x_2\ \mathrm{K}\ x_{n-1}]^{\mathrm{T}}$. Then the sliding condition, $s\dot{s} \le -\eta |s|$, is satisfied as $|s| \ge \bar{s}$, and the control system is stable in the sense that all system states x_i (i = 1, 2,..., n) are bounded by:

$$|x_i(t)| \le \left(2^{i-1} \Big/ \prod_{j=1}^{n-i} \lambda_j \right) \bar{s} \equiv \delta_i$$

For example, consider an underwater vehicle whose simplified model is represented as [11]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\frac{d}{m} x_2 |x_2| + \frac{1}{m} u \end{cases}$$

where x_1, x_2 represent the position error and velocity error of the vehicle, respectively; u is the control force; m is the mass of the vehicle; d denotes the drag coefficient. The parameter values that used in [11] are also adopted in the following simulations, *i.e.* $m = 3 + 1.5\sin(|x_2|t)$ and $d = 1.2 + 0.2\sin(|x_2|t)$.

Suppose that the weighting matrices are selected as $\underline{Q} = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}$, $R = 0.1$; the population size, the crossover rate and the mutation rate are selected as 10, 0.8 and 0.03, respectively. Based on [14], the optimal coefficients of sliding function can be derived as $\underline{c} = [1.4142\ \ 1]^{\mathrm{T}}$. Six fuzzy rules are created in this simulation. Fig. 1 shows the evaluation result of cost function and Fig. 2 shows the final state space response after learning.

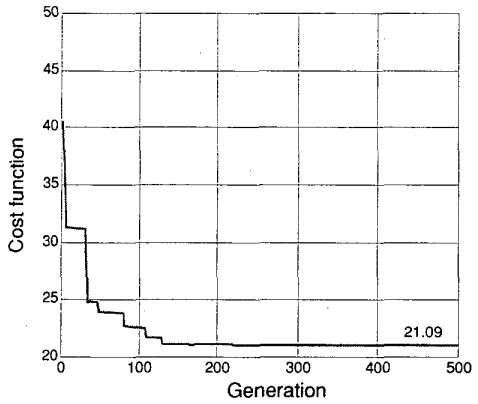

Fig. 1. The cost function

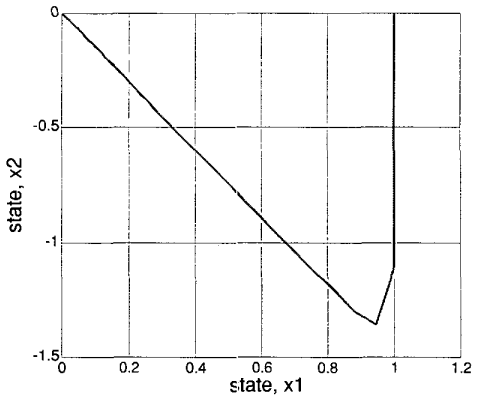

Fig. 2. The final state space response

CONCLUSION

In this paper, a new approach called modulus genetic algorithm was described. The numeric error, which arisen by the encoding/decoding procedure in conventional GAs, was avoided. In the modulus genetic algorithm, the encoding/decoding procedure is not necessary. It has the following advantages: 1) the evolution can be speeded up; 2) the numeric truncation error can be avoided; 3) the precision of solution can be increased.

The MGA was applied to resolve the key problem of fuzzy logic systems – rule acquisition. The fuzzy system with MGA forms an "intelligent fuzzy system". Based on the proposed learning step, the fuzzy rule base can be self-extracted and optimized. Such an intelligent fuzzy system has a general-purpose architecture. It can be applied to many kinds of fields, such as fuzzy control, fuzzy image processing, fuzzy decision making, and fuzzy pattern recognition, etc.

REFERENCES

1. Back, T., 1993. Optimal mutation rates in genetic search, Proc. 5th Int. Conf. on GAs, 2-8.
2. Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine learning, Addison-Wesley.
3. Goldberg, D.E., 1991. Real-coded genetic algorithms, virtual alphabets, and blocking, Complex Systems, 5, 129-167.
4. Karr, C.L., 1991. Applying genetics to fuzzy logic, AI Expert, March, 38-43.
5. Karr, C.L., 1991. Genetic algorithms for fuzzy controller, AI Expert, February, 26-33.
6. Lin, S.C., Chen, Y.Y., 1997. Design of self-learning fuzzy sliding mode controller based on genetic algorithms, Fuzzy Sets and Systems, 86, 139-153.
7. Tate, D.M., Smith, A.E., 1993. Expected allele coverage and the role of mutation in genetic algorithms, Int. Conf. on Genetic Algorithms, 31-37.
8. Wang, L.X., 1994. Adaptive Fuzzy Systems and Control, Englewood Cliffs, NJ: Prentice Hall.
9. Zadeh, L.A., 1965. Fuzzy sets, Information and Control, 8, 338-353.
10. Zadeh, L.A., 1973. Outline of a new approach to the analysis of complex systems and decision processes," IEEE Trans. on Systems, Man and Cybernetics, (SMC3), 28-44.
11. Lewis, F.L., 1992. Applied Optimal Control and Estimation, Englewood Cliffs, NJ: Prentice Hall.
12. Lin, S.C., Chen, Y.Y., 1997. Design of self-learning fuzzy sliding mode controller based on genetic algorithms, Fuzzy Sets and Systems, 86(2).
13. Lin, S.C., Chen, Y.Y., 1995. Design a hitting controller to stabilize the fuzzy sliding mode control systems, Proc. Int. Joint Conf. CFSA/IFIS/SOFT'95 - Fuzzy Theory and Apps., Taipei, 374-379.
14. Lin, S.C., 1997. Stable Self-Learning Optimal Fuzzy Control System Design and Application, Ph.D. Dissertation, National Taiwan University.