Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

Reinforcement Hybrid Evolutionary Learning for TSK-type Neuro-Fuzzy Controller Design

Yung-Chi Hsu, Sheng-Fuu Lin

Department of Electrical and Control Engineering
National Chiao-Tung University
1001 Ta Hsueh Road, Hsinchu, Taiwan 300, R.O.C.
(e-mail: sflin@mail.nctu.edu.tw)

Abstract: This paper proposes a recurrent TSK-type neuro-fuzzy controller (TNFC) with reinforcement hybrid evolutionary learning algorithm (R-HELA). The proposed R-HELA combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA) to perform the structure/parameter learning for constructing the TNFC dynamically. The evolution of a population consists of three major operations: group reproduction using the compact genetic algorithm, variable two-part crossover, and variable two-part mutation. Illustrative example is conducted to show the performance and applicability of the proposed R-HELA method.

1. INTRODUCTION

Recently, many evolutionary algorithms, such as the genetic algorithm (GA), genetic programming, evolutionary programming, and evolution strategies, have been proposed. Since they are heuristic and stochastic, they are less likely to get stuck at the local minimum, and they are based on populations made up of individuals with specific behaviours similar to certain biological phenomena. These common characteristics have led to the development of evolutionary computation as an increasingly important field.

The evolutionary fuzzy model generates a fuzzy system automatically by incorporating evolutionary learning procedures, where the well-known procedure is GA. Several genetic fuzzy models, that is, fuzzy models augmented by a learning process based on GAs, have been proposed. Karr applied GAs to the design of the membership functions of a fuzzy controller (Karr 1991), with the fuzzy rule set assigned in advance. Since the membership functions and rule sets are co-dependent, simultaneous design of these two approaches would be a more appropriate methodology. Bandyopadhyay *et. al.* used the variable-length genetic algorithm (VGA) that let the different lengths of the chromosomes in the population (Bandyopadhyay et al. 2000). Juang *et. al.* proposed genetic reinforcement learning in design of fuzzy controllers (Juang et al. 2000). The GA that Juang *et. al.* adopted was based upon traditional symbiotic evolution which, when applied to fuzzy controller design, complements the local mapping property of a fuzzy rule. However, the aforementioned approaches may require one or more of the following: 1) the number of fuzzy rules has to be assigned in advance; 2) the lengths of the chromosomes in the population must be the same.

For solving above problems, in this paper, we present a TSK-type neuro-fuzzy controller (TNFC) with a reinforcement hybrid evolutionary learning algorithm (R-HELA). The proposed R-HELA determines the number of fuzzy rules automatically and processes the variable-length chromosomes. The length of each individual denotes the total number of genes in that individual. The initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Individuals with an equal number of rules constitute the same group. Thus, initially there are several groups in a population. We use the elite-based reproduction strategy to keep the best group. Therefore, the best group can be reproduced many times for each generation. The reinforcement signal from the environment is used as a fitness function for the R-HELA. That is, we formulate the number of time steps before failure occurs as the fitness function. In this way, the R-HELA can evaluate the candidate solutions for the parameters of the TNFC model.

The advantages of the proposed R-HELA method are summarized as follows: 1) it determines the number of fuzzy rules and tune the free parameters of the TNFC model in a highly autonomous way. Thus, users need not give it any *a priori* knowledge or even any initial information on these. 2) It is applicable to chromosomes of different lengths. 3) It does not require precise training data for setting the parameters of the TNFC model. 4) It is indeed that the algorithm can perform better and converge more quickly than some traditional genetic methods.

This paper is organized as follows. In section 2, the TSK-type neuro-fuzzy controller (TNFC) is introduced. In section 3, the proposed hybrid evolution learning algorithm (HELA) is described. In section 4, the reinforcement hybrid evolution learning algorithms used for constructing the TNFC controller is introduced. In section 5, the simulation results are presented. The conclusions are summarized in the last section.

2. TSK-TYPE NEURO-FUZZY CONTROLLERS

A Takagi-Sugeno-Kang (TSK) type controller (Lin et al. 1996) employs different implication and aggregation methods than the standard Mamdani controller. Instead of using fuzzy sets, the conclusion part of a rule is a linear combination of the crisp inputs.

$$\textbf{IF } x_1 \textbf{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textbf{ and } x_2 \textbf{ is } A_{2j}(m_{2j}, \sigma_{2j})\ldots \textbf{ and } x_n \textbf{ is } A_{nj}(m_{nj}, \sigma_{nj}) \textbf{ THEN } y' = w_{0j} + w_{1j}x_1 + \ldots + w_{nj}x_n \tag{1}$$

Since the consequence of a rule is crisp, the defuzzification step becomes obsolete in the TSK inference scheme. Therefore, the controller's output is computed as the weighted average of the crisp rule outputs, which is computationally less expensive then calculating the center of gravity.

In this paper, we adopt a TSK-type neuro-fuzzy controller (TNFC) to perform a control problem. The functions of the nodes in each layer are described as follows:
Layer1 *(Input Node):* No function is performed in this layer. The node only transmits input values to layer 2.

$$u_i^{(1)} = x_i \tag{2}$$

Layer2 *(Membership Function Node)****:*** Nodes in this layer correspond to one linguistic label of the input variables in layer1; that is, the membership value specifying the degree to which an input value belongs to a fuzzy set is calculated in this layer. For an external input x_i, the following Gaussian membership function is used:

$$u_{ij}^{(2)} = \exp\left(-\frac{\left[u_i^{(1)} - m_{ij}\right]^2}{\sigma_{ij}^2}\right) \tag{3}$$

where m_{ij} and σ_{ij} are, respectively, the center and the width of the Gaussian membership function of the *j*th term of the *i*th input variable x_i.
Layer 3 *(Rule Node)****:*** The output of each node in this layer is determined by the fuzzy AND operation. Here, the product operation is utilized to determine the firing strength of each rule. The function of each rule is

$$u_j^{(3)} = \prod_i u_{ij}^{(2)} \tag{4}$$

Layer 4 *(Consequent Node)****:*** Nodes in this layer are called consequent nodes. The input to a node in layer 4 is the output delivered from layer 3, and the other inputs are the input variables from layer 1. For this kind of node, we have

$$u_j^{(4)} = u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i) \tag{5}$$

where the summation is over all the inputs and w_{ij} are the corresponding parameters of the consequent part. The w_{ij} is any real value. If $w_{ij}=0$, $i>0$, the TNFC controller in this case will be called the zero-order TNFC controller.
Layer 5 *(Output Node)****:*** Each node in this layer corresponds to one output variable. The node integrates all the actions recommended by layers 3 and 4 and acts as a defuzzifier with

$$y = u^{(5)} = \frac{\sum_{j=1}^{M} u_j^{(4)}}{\sum_{j=1}^{M} u_j^{(3)}} = \frac{\sum_{j=1}^{M} u_j^{(3)}(w_{0j} + \sum_{i=1}^{n} w_{ij}x_i)}{\sum_{j=1}^{M} u_j^{(3)}} \tag{6}$$

where M is the number of fuzzy rules.

3. A HYBRID EVOLUTION LEARNING ALGORITHM

In this section, the proposed hybrid evolutionary learning algorithm (HELA) will be introduced. Recently, many efforts that try to enhance the traditional GAs have been made (Michalewicz 1999). Among them, one category focuses on modifying the structure of a population or the role an individual plays in it, such as the distributed GA (Tanese 1989), the cellular GA (Arabas et al. 1994), and the symbiotic GA (Moriarty et al. 1996).

In a traditional evolution algorithm, the number of rules in a model must be predefined. Our proposed HELA combines the compact genetic algorithm (CGA) and the modified variable-length genetic algorithm (MVGA). In the MVGA, the initial length of each individual may be different from each other, depending on the total number of rules encoded in it. Thus, we do not need to predefine the number of rules. In this paper, individuals with an equal number of rules constitute the same group. Initially, there are several groups in a population. Unlike the traditional variable-length genetic algorithm (VGA) (Bandyopadhyay et al. 2000), Bandyopadhyay *et. al.* used "#" to mean, "does not care". In this study, we adopt the variable two-part crossover (VTC) and the variable two-part mutation (VTM) to make the traditional crossover and mutation operators applicable to different lengths of chromosomes. Therefore, we do not use "#" to mean, "does not care" in the VTC and the VTM.

In this study, we divide a chromosome into two parts. The first part of the chromosome gives the antecedent parameters of a TNFC model while the second part of the chromosome gives the consequent parameters of a TNFC model. Each part of the chromosome can be performed using the VTC on the overlapping genes of two chromosomes. In the traditional VGA, Bandyopadhyay *et. al.* only evaluated the performance of each chromosome in a population. The performance of the number of rules was not evaluated in (Bandyopadhyay et al. 2000). In this study, we use the elite-based reproduction strategy to keep the best group with the same length chromosomes. Therefore, the best group can be reproduced many times for each generation. The elite-based reproduction strategy is similar to the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire knowledge from society.

In the proposed HELA method, we adopt the compact genetic algorithm (CGA) (Harik et al. 1999) to carry out the elite-based reproduction strategy. The CGA represents a population as a probability distribution over the set of solutions and is operationally equivalent to the order-one behavior of the simple GA (Lee et al. 1995). The advantage of the CGA is that it processes each gene independently and requires less memory than the normal GA. The building blocks (BBs) in the CGA represent the suitable lengths of the chromosomes and reproduce the chromosomes according to the BBs. The coding scheme consists of the coding done by the MVGA and the CGA. The MVGA codes the adjustable parameters of a TNFC model into a chromosome, as shown in Fig. 1; where MS$_j$ represents the parameters of the antecedent of the *j*th rule in the TNFC, C$_j$ represents the parameters of the consequent of the *j*th rule. In Fig. 2, the CGA codes the probability vector into the building blocks (BBs), where each probability vector represents the suitability of the rules of a TNFC model. In CGA, we must predefine the maximum number of rules (M_{max}) and the minimum number of rules (M_{min}) to prevent the number of fuzzy rules generated beyond a certain bound (i.e., [M_{max}, M_{min}]).

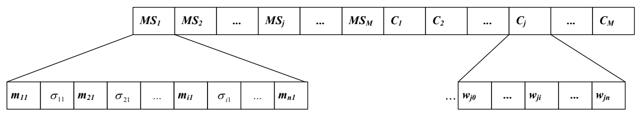

Fig. 1. Coding the adjustable parameters of a TNFC into a chromosome in the MVGA.

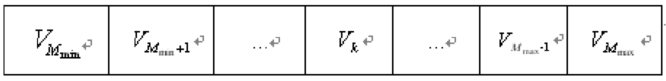

Fig. 2. Coding the probability vector into the building blocks (BBs) in the CGA.

The learning process of the HELA involves three major operators: elite-based reproduction strategy, variable two-part crossover, and variable two-part mutation. The major learning process is described step-by-step as follows:

***a.* Elite-Based Reproduction Strategy (ERS):** Reproduction is a process in which individual strings are copied according to their fitness value. A fitness value is assigned to each individual using Eqs. (13). The goal of the R-HELA method is to maximize the fitness value. The higher a fitness value, the better the fitness. In this study, we use an elite-based reproduction strategy (ERS) to mimic the maturing phenomenon in society, where individuals become more suitable to the environment as they acquire more knowledge from society. The CGA is used here to perform the ERS. The CGA represents the population as a probability distribution over the set of solutions and is operationally equivalent to the order-one behavior of the simple GA. The CGA uses the BBs to represent the suitable length of the chromosomes and reproduces the chromosomes according to the probability vector in the BBs. The best performing individuals where in the top half of each population are using to perform the ERS. According to the results of the ERS, using the crossover and the mutation operations generates the other half individuals. After the ERS, the suitable length of chromosomes will be preserved and the unsuitable length of chromosomes will be removed. The detailed of the ERS is shown as follows:

Step 1. Update the probability vectors of the BBs according to the following equations:

$$\begin{cases} V_k = V_k + (Upt\_value_k * \lambda), & if \quad Avg \le Max\_fit_k \\ V_k = V_k - (Upt\_value_k * \lambda), & otherwise \end{cases} \tag{7}$$
$$where\ k=[M_{max}, M_{min}]$$

$$Avg = \sum_{p=1}^{Nc} fit_p / Nc \tag{8}$$

$$Upt\_value_k = Total\_fit_k \Big/ \sum_{p=1}^{Nc} fit_p \tag{9}$$

$$Total\_fit_k = \sum_{p=1}^{N_k} fit_{kp} \tag{10}$$

where V_k is the probability vector in the BBs and represents the suitable chromosome in the group with k rules in a population; λ is a threshold value we predefine; Avg represents the average fitness value in the whole population; Nc is the population size; N_k is the kth group size; fit_p is the fitness value of the pth chromosome in all Nc populations; fit_{kp} is the fitness value of the pth chromosome in kth group; and Max_fit_k is the best fitness value (maximum value of Eq. (13)) in the kth group. As shown in Eq. (7), if $Max\_fit_k \ge Avg$, then the suitable chromosomes in the kth group should be increased. On the other hand, if $Max\_fit_k < Avg$, then the suitable chromosomes in the kth group should be decreased. Eq. (10) represents the sum of the fitness values of the chromosomes in the kth group.

Step 2. Determine the reproduction number according to the probability vectors of the BBs as follows:

$$Rep_k = (P_{size}/2) * (V_k / Total\_Velocy)$$
$$where\ k=[M_{max}, M_{min}] \tag{11}$$
$$Total\_Velocy = \sum_{k=R_{\min}}^{M_{\max}} V_k \tag{12}$$

where P_{size} represents the population size; Rep_k is the recorder, and a chromosome has k rules for constructing a TNFC.

Step 3. After step 2, the reproduction number of each group in the top half of a population is obtained. Then we generate Rep_k chromosomes each group using the roulette-wheel selection method (Cordon et al. 2001).

Step 4. If any probability vector in BBs reaches 1, then stop the ERS and set the probability vector to 1 for all groups with the same number of rules, according to step 2. The lacks of the chromosomes are generated randomly. To replace the ERS step, we use the roulette-wheel selection method (Cordon et al. 2001) – a simulated roulette is spun – for this reproduction process.

b. Variable two-part crossover: Although the ERS operation can search for the best existing individuals, it does not create any new individuals. In nature, an offspring has two parents and inherits genes from both. The main operator working on the parents is the crossover operator, the operation of which occurs for a selected pair with a crossover rate. In this paper, we propose the variable two-part crossover (VTC) to perform this step. In the VTC, the parents are selected from the enhanced elites. In the VTC, two parents are selected using the roulette-wheel selection method (Cordon et al. 2001). The two parents may be selected from the same or different groups. Performing crossover on the selected parents creates the offspring. Since the parents may be of different lengths, we must avoid misalignment of individuals in the crossover operation. Therefore, a variable two-part crossover is proposed to solve this problem. The first part of the chromosome gives the antecedent parameters of a TNFC model while the second part of the chromosome gives the consequent parameters of a TNFC model. The two-point crossover is adopted in each part of the chromosome. Thus, new individuals are created by exchanging the site's values between the selected sites of the parents' individuals. To avoid the misalignment of individuals in the crossover operation, in the VTC, the selection of the crossover points in each part will not exceed the shortest length chromosome of two parents. Two individuals of different lengths using the variable two-part crossover operation are shown in Fig. 3. MS_j represents the parameters of the antecedent part of the *j*th rule in the TNFC, W_j represents the parameters of the consequent of the *j*th rule in the TNFC, and M_k is the number of fuzzy rules in *k*th chromosome. After the VTC operation, the individuals with poor performance are replaced by the new offspring.

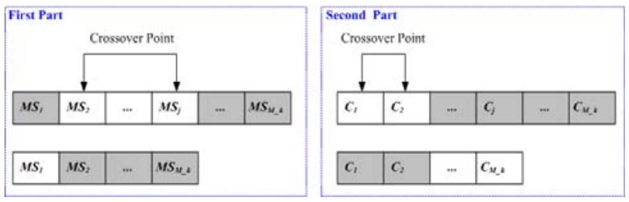

Fig. 3. The VTC in the HELA.

c. Variable two-part mutation: In this paper, the variable two-part mutation (VTM) is proposed to perform the mutation operation. In each part of a chromosome, uniform mutation is adopted, and the mutated gene is drawn randomly from the domain of the corresponding variable.

4. REINFORCEMENT LEARNING FOR A TNFC

Unlike the supervised learning problem, in which the correct "target" output values are given for each input pattern, the reinforcement learning problem has only very simple "evaluative" or "critical" information, rather than "instructive" information, available for learning. In the extreme case, there is only a single bit of information to indicate whether the output is right or wrong. The reinforcement hybrid evolutionary learning algorithm (R-HELA) and its training environment interact in a reinforcement learning problems are shown in Fig. 4. In this paper, the reinforcement signal indicates whether a success or a failure occurs.

As show in Fig. 4, the proposed R-HELA consists of a TNFC model, which acts as the control network to determine a proper action according to the current input vector (environment state). The structure of the proposed R-HELA is different from Barto and his colleagues' actor-critic architecture (Barto et al. 1983), which consists of a control network and a critic network. The input to the TNFC model is the state of the plant, and the output is a control action of the state, denoted by *f*. The only available feedback is a reinforcement signal that notifies the TNFC model only when a failure occurs. An accumulator plays a role which is a relative performance measure shown in Fig. 4. It accumulates the number of time steps before a failure occurs. In this paper, the feedback takes the form of an accumulator that determines how long the experiment is still a "success"; this is used as a relative measure of the fitness of the proposed R-HELA method. That is, the accumulator will indicate the "fitness" of the current TNFC model. The key to the R-HELA is formulating a number of time steps before failure occurs and using this formulation as the fitness function of the R-HELA method.

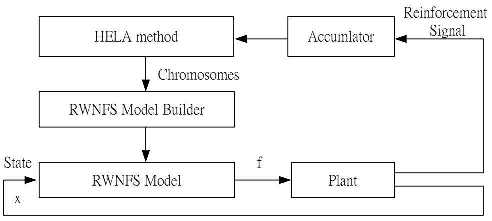

Fig. 4. Schematic diagram of the R-HELA for the TNFC.

In this paper, we use a number of time steps before failure occurs to define the fitness function. The goal of the R-HELA method is to maximize the fitness value. The fitness function is defined by:

$$\textit{Fitness Value (i)} = \textit{TIME-STEP(i)} \tag{13}$$

where *TIME-STEP(i)* represents how long the experiment is a "success" with the *i*th population.

5. ILLUSTRATIVE EXAMPLE

In this section, we compare the performance of the TNFC model using the R-HELA method with an application. The simulation was performed to balance the cart-inverted-pendulum plant (Cheok et al. 1987). The initial parameters are given in Table 1. The initial parameters are determined by practical experimentation or trial-and-error tests.

Table 1: The initial parameters before training

Parameters	*Value*	*Parameters*	*Value*
Population Size	*54*	$[w_{\min}, w_{\max}]$	*[-20,20]*
Crossover Rate	*0.5*	$M_{\max}$	*12*
Mutation Rate	*0.3*	$M_{\min}$	*3*
$[\sigma_{\min}, \sigma_{\max}]$	*[0,2]*	λ	*0.01*
$[m_{\min}, m_{\max}]$	*[0,2]*		

In this example, we shall apply the R-HELA method to the classic control problem of the cart-inverted-pendulum plant. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and the classic control techniques (Cheok et al. 1987), or the reinforcement learning schemes (Barto et al. 1983), and is now used as a control benchmark. The cart-inverted-pendulum plant is the problem of learning how to balance an upright pole. The bottom of the pole is hinged to a cart that travels along a finite-length track to its right or left. Both the cart and the pole can move only in the vertical plane; that is, each has only one degree of freedom.

There are four state variables in the system: θ, the angle of the pole from an upright position (in degrees); $\dot{\theta}$, the angular velocity of the pole (in degrees/seconds); x, the horizontal position of the cart's center (in meters); and $\dot{x}$, the velocity of the cart (in meters/seconds). The only control action is *f*, which is the amount of force (in *Newtons*) applied to cart to move it toward left or right. The system fails when the pole falls past a certain angle (± 24 is used here) or the cart runs into the bounds of its track (the distance is 2.4 m from the center to each bound of the track). The goal of this control problem is to determine a sequence of forces applying to the cart to balance the pole upright. The equations of motion that we used are:

$$\theta(t+1)=\theta(t)+\Delta\dot{\theta}(t) \tag{14}$$

$$\dot{\theta}(t+1)=\dot{\theta}(t)+\Delta\frac{(m+m_p)g\sin\theta(t)}{(4/3)(m+m_p)l-m_pl\cos^2\theta(t)} \tag{15}$$

$$-\frac{\cos\theta(t)\left[f(t)+m_pl\dot{\theta}(t)^2\sin\theta(t)-M_c\,\mathrm{sgn}(\dot{x}(t))\right]}{(4/3)(m+m_p)l-m_pl\cos^2\theta(t)}$$

$$-\frac{\dfrac{\mu_p(m+m_p)\dot{\theta}(t)}{m_pl}}{(4/3)(m+m_p)l-m_pl\cos^2\theta(t)}$$

$$x(t+1)=x(t)+\Delta\dot{x}(t) \tag{16}$$

$$x(t+1)=\dot{x}(t)+\Delta\frac{f(t)+m_pl\left[\dot{\theta}(t)^2\sin\theta(t)-\ddot{\theta}(t)\cos\theta(t)\right]}{(m+m_p)} \tag{17}$$

$$-\frac{\mu_c\,\mathrm{sgn}(\dot{x}(t))}{(m+m_p)}$$

where

$l = 0.5$ m, the length of the pole;
m = 1.1 kg, combined mass of the pole and the cart;
m_p = 0.1 kg, mass of the pole;
g = 9.8 m/s , acceleration due to the gravity; (18)
M_c = 0.0005, coefficient of friction of the cart on the track,
M_p = 0.000002, coefficient of friction of the pole on the cart,
Δ = 0.02(s), sampling interval.

The constraints on the variables are $-24^\circ \le \theta \le 24^\circ$, -2.4m$\le x \le$ 2.4m, and -10N$\le f \le$ 10N. A control strategy is deemed successful if it can balance a pole for 100000 time steps. The four input variables $(\theta,\dot{\theta},x,\dot{x})$ and the output f_t are normalized between 0 and 1. The four normalized state variables are used as inputs to the proposed TNFC model. The fitness function in this example is defined in Eq. (13) to train the TNFC model. A total of thirty runs were performed. Each run started at the different initial state. The TNFC model learned to balance the pole at the 54*th* generation averagely is shown in Fig. 5. In this figure, each run represents that largest fitness value in the current generation is selected before the cart-pole balancing system fails. When the R-HELA method is stopped, we choose the best strings in the population at the final generation and test them on the cart-inverted-pendulum plant. Fig. 6 shows the results of the probability vectors in CGA. In this figure, the final average optima number of rules is 4.

The simulation was carried out for thirty runs. The successful results, which consist of the pole angle, cart position and controller output, are shown in Fig. 5 (d)-(f). Each line in Fig. 5 (d)-(f) represents each run with a different initial state. The results shown in this figure are the first 1000 time steps in the 100,000 control time steps. As shown in Fig. 5 (d)-(f), the R-HELA successfully controlled the cart-inverted-pendulum plant in thirty runs.

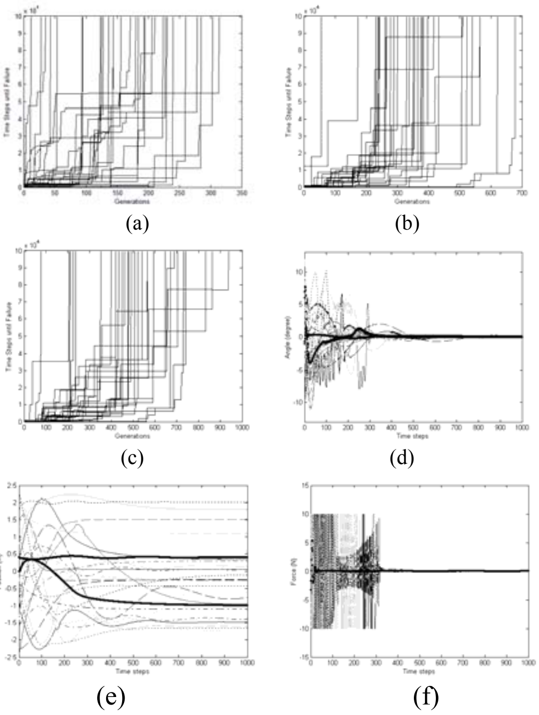

Fig. 5. The performance of time steps of (a) the R-HELA, (b) the R-SE, and (c) the R-GA and control results of the cart-inverted-pendulum plant using the R-HELA of (d) angle of the pole, (e) position of the cart, and (f) control force.

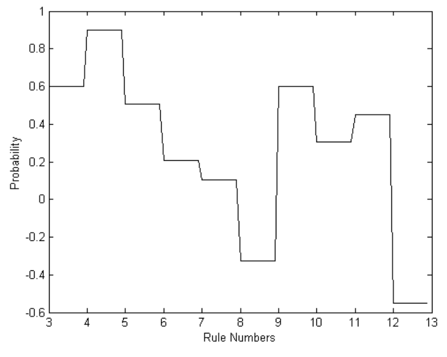

Fig. 6. The probability vectors of ERS step in R-HELA.

We also compare the performance of our system with the reinforcement symbiotic evolution (R-SE) (Juang et al. 2000) and the reinforcement genetic algorithm (R-GA) (Karr 1991) was applied to the same problem. In the R-GA and the R-SE, the population size was set to 200 and the crossover and mutation probabilities were set to 0.5 and 0.3, respectively. The R-SE and R-GA methods learned to balance the pole at the 80*th* and 149*th* generations averagely are shown in Fig. (b) and (c).

The GENITOR (Whitley et al. 1993), the SANE (Symbiotic Adaptive Neuro-Evolution) (Moriarty et al. 1996), the TDGAR (Lin et al. 2000), and the CQGAF (Juang 2005) have been applied to the same control problem and the simulation results are listed in Table 2. Table 2 shows the number of pole-balance trials (which reflects the number of training episodes required) and CPU time. In this experiment, we used a Pentium 4 chip with a 1.5GHz CPU, a 512MB memory, and the visual C++ 6.0 simulation software. As shown in Table 2, the proposed R-HELA is feasible and effective and obtains smaller CPU times than those of other existing models.

Table 2. Performance comparison of various existing models.

Method	Mean	Mean (time)	Best	Best (time)	Worst	Worst (time)
Whitley et al. 1993	3814	104.65	519	51.68	9172	218.51
Moriarty et al. 1996	2148	76.25	89	38.56	5482	178.34
Karr 1991	514	72.34	78	33.56	938	130.75
Juang et al. 2000	346	68.37	56	28.72	679	121.39
Lin et al. 2000	287	61.34	49	21.78	512	116.91
Juang 2005	213	52.92	47	18.67	489	107.64
R-HELA	**198**	**28.47**	**12**	**6.31**	**314**	**81.83**

6. CONCLUSIONS

In this paper, a TSK-type neuro-fuzzy controller (TNFC) with the reinforcement hybrid evolutionary learning algorithm (R-HELA) is proposed for dynamic control problems. The proposed R-HELA has structure-and-parameter learning ability. That is, it can determine the average optima number of fuzzy rules and tune the free parameters in the TNFC. The proposed learning method also processes variable lengths of the chromosomes in a population. The computer simulation has shown that the proposed R-HELA has a better performance than the other methods.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Council, Taiwan. R. O. C. under Grant NSC 95-2221-E-009-214 and NSC 95-2752-E-009-011-PAE.

REFERENCES

Arabas J., Michalewicz Z., and Mulawka J. (1994). GAVaPS—A Genetic Algorithm with Varying Population Size. *Proceedings of IEEE International Conference on Evolutionary Computation:* 73–78, 1994.

Bandyopadhyay S., Murthy C. A., and Pal S. K. et al. (2000). VGA-classfifer: Design and Applications. *IEEE Transaction on System, Man, and Cybernetics. Part B: Cybernetics.* 30: 890–895.

Barto A. G., Sutton R. S., and Anderson C. W. (1983). Neuron Like Adaptive Elements that can Solve Difficult Learning Control Problem. *IEEE Transaction on System, Man, and Cybernetics.* 13 (5): 834-847.

Cordon O., Herrera F., Hoffmann F., and Magdalena L. (2001). *Genetic Fuzzy Systems Evolutionary Tuning and Learning of Fuzzy Knowledge Bases.* Advances in Fuzzy Systems-Applications and Theory. 19. NJ: World Scientific Publishing.

Cheok K. C. and Loh N. K. (1987). A Ball-Balancing Demonstration of Optimal and Disturbance-Accommodating Control. *IEEE Contr. Syst. Mag.* 54–57.

Harik G.R., Lobo F.G. and Goldberg D.E. (1999). The Compact Genetic Algorithm. *IEEE Transactions* on *Evolutionary Computation*. 3 (4): 287-297.

Juang C. F., Lin J. Y. and Lin C. T. (2000). "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Trans on System, Man, and Cybernetics. Part B: Cybernetics.* 30 (2): 290-302.

Juang C. F. (2005). Combination of Online Clustering and Q-value based GA for Reinforcement Fuzzy System design. *IEEE Transactions on Fuzzy Systems.* 13 (3): 289–302.

Karr C. L. (1991). Design of an Adaptive Fuzzy Logic Controller using a Genetic Algorithm. *Proceedings of the Fourth International Conference on Genetic Algorithms*: 450–457.

Lin C. T. and Lee C. S. G. et al. (1996). *Neural Fuzzy Systems: A Neuro-Fuzzy Synergism to Intelligent System*, NJ:Prentice-Hall.

Lee K.Y., Xiaomin B., and Park Y. M. (1995). Optimization method for reactive power planning by using a modified simple genetic algorithm. *IEEE Trans.on Power Systems.* 10 (4): 1843-1850.

Lin C. T. and Jou C. P. (2000). GA-based Fuzzy Reinforcement Learning for Control of a Magnetic Bearing System. *IEEE Transaction on System, Man, and Cybernetics. Part B: Cybernetics.* 30 (2): 276-289.

Michalewicz Z. (1999) Genetic Algorithms+Data Structures=Evolution Programs. New York: Springer-Verlag.

Moriarty D. E. and Miikkulainen R. (1996). Efficient Reinforcement Learning through Symbiotic Evolution. *Mach. Learn.*, 22: 11–32, 1996.

Tanese R. (1989). Distributed Genetic Algorithm. *Proceedings of International Conference on. Genetic Algorithms*: 434–439.

Whitley D., Dominic S., Das R., and Anderson C. W. (1993). Genetic Reinforcement Learning for Neuro Control Problems. *Mach. Learn.* 13: 259–284., 1993.