20

Symbiotic Evolution Genetic Algorithms for Reinforcement Fuzzy Systems Design

Chia-Feng Juang* and I-Fang Chung**
**Department of Electrical Engineering, National Chung-Hsing University*
***Institute of Biomedical Informatics, National Yang-Ming University*
Taiwan, R.O.C.

1. Introduction

The advent of fuzzy logic controllers has inspired the allocation of new resources for the possible realization of more efficient methods of control. In comparison with traditional controller design methods requiring mathematical models of the plants, one key advantage of fuzzy controller design lies in its model-free approach. Conventionally, the selection of fuzzy if-then rules often relies heavily upon the substantial amounts of heuristic observation to express the strategy's proper knowledge. It is very difficult for human experts to examine all the input-output data from a complex system, and then to design a number of proper rules for the fuzzy logic controllers. Many design approaches for automatic fuzzy rules generation have been developed in an effort to tackle this problem (Lin & Lee, 1996). The neural learning method is one of them. In (Miller et al., 1990), several neural learning methods including supervised and reinforcement based control configurations are studied. For many control problems, the training data are usually difficult and expensive, if not impossible, to obtain. Besides, many control problems require selecting control actions whose consequences emerge over uncertain periods for which training data are not readily available. In reinforcement learning, agents learn from signals that provide some measure of performance which may be delivered after a sequence of decisions being made. Hence, when the above mentioned control problems occur, reinforcement learning is more appropriate than supervised learning.

Genetic algorithms (GAs) are stochastic search algorithms based on the mechanics of natural selection and natural genetics (Goldberg, 1989). Since GAs do not require or use derivative information, one appropriate application for their use is the circumstance where gradient information is unavailable or costly to obtain. Reinforcement learning is an example of such domain. The link of GAs and reinforcement learning may be called genetic reinforcement learning (Whitley et al., 1993). In genetic reinforcement learning, the only feedback used by the algorithm is the information about the relative performance of different individuals and may be applied to reinforcement problems where the evaluative signals contain relative performance information. Besides GAs, another general approach for realizing reinforcement learning is the temporal difference (TD) based method (Sutton & Barto, 1998). One generally used TD-based reinforcement learning method is Adaptive Heuristic Critic (AHC) learning algorithm. AHC learning algorithm relies upon both the learned evaluation

Source: Advances in Evolutionary Algorithms, Book edited by: Witold Kosiński, ISBN 978-953-7619-11-4, pp. 468, November 2008, I-Tech Education and Publishing, Vienna, Austria

network and the learned action network. Learning of these two networks is based on gradient-descent learning algorithms with errors derived from internal and external reinforcement signals. In comparison with the GAs, one disadvantage of AHC learning algorithms is that they usually suffer the local minimum problem in network learning due to the use of the gradient descent method. Overall performance comparisons between TD-based reinforcement learning methods, including AHC and Q-learning, and GAs are made in (Whitley et al., 1993; Moriarty & Miikkulainen, 1996). The results show that GAs achieve better performance both in CPU time and number of control trials. In the past, some studies on the combination of GAs with TD-based reinforcement learning methods were proposed (Lin & Jou, 1999; Juang, 2005a). These studies show that the combination approach achieves better performance than using only GAs or the TD-based method.

Many approaches to fuzzy system design using GAs have been proposed (Cordón et al., 2004). If we distinguish them by individual representation in GAs, the major ones include Pittsburgh, Michigan, and the iterative rule learning (IRL) approach (Cordón et al., 2001). In the Pittsburgh approach, each individual represents an entire fuzzy rule set. A population of candidate rule sets is maintained by performing genetic operators to produce new generations of rule sets. Most GA-based fuzzy controller design methods belong to this approach (Karr, 1991; Homaifar & McCormick, 1995; Shi et al., 1999; Belarbi & Titel, 2000; Chung et al., 2000; Juang, 2004; Chou, 2006). In (Karr, 1991), Karr applied GAs to the design of the membership functions of a fuzzy controller, 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. Based upon this concept, many researchers have applied GAs to optimize both the parameters of the membership functions and the rule sets. Differences between the approaches depend mainly on the type of coding and the way in which the membership functions are optimized. The disadvantage of this approach is the computational cost, since a population of rule set has to be evaluated in each generation. Also, the dimension of search space increases significantly, making it substantially difficult to find good solutions. In the Michigan approach, each individual of the population represents a single rule and a rule set is represented by the entire population. All researches in (Valenzuela-Rendon, 1991; Bonarini, 1993; Furuhashi et al., 1995) belong to this approach. As the evolutionary process is applied to the individual rule base, this approach invariably leads to consideration of both cooperation and competition. Obviously, it is difficult to obtain a good cooperation among the fuzzy rules that compete with each other. To solve this cooperation versus competition problem, a complex credit assignment policy is required, which is a disadvantage of this approach. This credit assignment task becomes more difficult especially for controller design based upon reinforcement learning problems, where the reinforcement signal is available after a long sequence of control actions. Besides, if the rule number in the designed fuzzy system is small, the small rule set in the population may easily converge to a local optimum and degrade the search speed. Like the Michigan approach, in IRL, each individual represents a single rule. However, in contrast to the former, only the best rule is adopted and added to the rule set in every GA run. The process is run several times to obtain the complete rule set. The IRL approach is considered to design genetic processes for off-line inductive learning problems and is not suitable to the controller design problem considered here.

Recently, the adoption of coevolutionary GAs for fuzzy system design has also been proposed. In GAs, coevolution refers to the simultaneous evolution of two or more species

with coupled fitness (Zhao, 1998; Paredis, 1995). Coevolution may be performed at the population level (Potter et al., 1995; Peña -Reyes & Sipper, 2001) or at the individual level (Juang et al., 2000). The idea of coevolutionary GA is similar to the Michigan approach. The cooperative coevolutionary GAs (Potter & DeJong, 2000; Peña -Reyes & Sipper, 2001) and Symbiotic GAs (Moriarty & Miikkulainen, 1996; Moriarty & Miikkulainen, 1998; Juang et al., 2000; Juang, 2005b; Lin & Xu, 2006) are of this type. In (Moriarty & Miikkulainen, 1996; Moriarty & Miikkulainen, 1998), Symbiotic Adaptive Neuro-Evolution (SANE) and hierarchical SANE were proposed for neural networks design. In (Juang et al., 2000), a Symbiotic-Evolution-based Fuzzy Controller (SEFC) was proposed and the performance of SEFC is shown to be better than SANE. In (Juang, 2005b), a coevolutionary GA with divide-and-conquer (CGA-DC) technique was proposed. The CGA-DC not only performs a GA search on separate fuzzy rules, but also on a global fuzzy network simultaneously. Therefore, the performance of CGA-DC is better than SEFC. This chapter extends the idea of CGA-DC to both feedforward and recurrent fuzzy systems design, and the design method is called hierarchical SEFC (HSEFC).

Besides GAs, another factor that may influence fuzzy controller performance is its structure. Depending on the property of a controlled plant, different types of fuzzy controller structures are used in this chapter. A feedforward fuzzy controller is designed for a static plant. For a dynamic plant, whose output depends upon either previous states or control inputs or both, a recurrent controller should be a better choice. To apply a feedforward controller to this type of plant, we need to know the exact order of the plant in advance, and the inclusion of the past values to the controller input increases the controller size. Several recurrent fuzzy systems have been proposed (Zhang & Morris, 1999; Juang & Lin, 1999; Lee &. Teng, 2000; Juang, 2002). The performance of these systems has been demonstrated to be superior to that of recurrent neural networks. Based on this observation, a recurrent fuzzy controller should be a better choice compared to a recurrent neural controller under genetic reinforcement learning.

This Chapter introduces feedforward and recurrent fuzzy controllers design using HSEFC. For a static plant control problem under reinforcement learning environment, HSEFC for feedforward fuzzy controller design (HSEFC-F) is introduced, while for a dynamic plant, HSEFC for recurrent fuzzy controller (HSEFC-R) is proposed. In HSEFC-F, two populations are created. One of the populations is for searching the well-performed local rules, and each individual in the population represents only a fuzzy rule. Within the other population, each individual represents a whole fuzzy controller. The objective of the population is to search the best fuzzy system participating rules selected from the rule population, and the relationship between each rule is cooperative. Concurrent evolution of the local-mapping and global-mapping stages increases the design efficiency. With the above techniques, HSEFC-F performs an efficient fuzzy controller design task with a small population size. HSEFC-R is applied to the design of a recurrent fuzzy controller obtained by adding feedback structures into the feedforward fuzzy systems. In the local-mapping stage, each recurrent fuzzy rule is divided into two sub-rules, one representing a spatial mapping and the other doing a temporal mapping. These two sub-rules are considered as two distant species, and two populations are created for each sub-rule search, which is a technique based on the divide-and-conquer concept. In the global-mapping search stage, the third population is created to seek the best combination of spatial sub-rules, temporal sub-rules or both.

This chapter is organized as follows. Section 2 describes the types and functions of the fuzzy controller to be designed, including feedforward and recurrent fuzzy controllers. Section 3 describes the concepts of symbiotic evolution for fuzzy systems. Section 4 introduces HSEFC for fuzzy controller design, including HSEFC-F and HSEFC-R. Section 5 presents simulation results, where HSEFC-F is applied to control a cart-pole balancing system and HSEFC-R is applied to control a dynamic system with delays. Comparisons with SEFC for the same task are also made in this section. The conclusions are summarized in the last section.

2. Fuzzy controller

Control Systems represent an important application for reinforcement learning algorithms. From the perspective of controller learning, since GAs only require the appropriate evaluation of the controller performance to yield the fitness values for evolution, they are suitable for fuzzy controller design under reinforcement learning problems.

2.1 Feedforward fuzzy controller

Several types of fuzzy systems have been proposed depending on the types of fuzzy if-then rules and fuzzy reasoning. In this chapter, each rule in the feedforward fuzzy controller is presented in the following form:

$$\text{Rule } i: \text{ IF } x_1(t) \text{ is } A_{i1} \text{ And } \ldots \text{ And } x_n(t) \text{ is } A_{in} \text{ Then } u(t+1) \text{ is } b_i \tag{1}$$

where x_j is the input variable, u is the control output variable, A_{ij} is a fuzzy set, and b_i is a fuzzy singleton. For a fuzzy set A_{ij}, a Gaussian membership function with

$$M_{ij}(x_j) = \exp\{-(\frac{x_j - m_{ij}}{\sigma_{ij}})^2\} \tag{2}$$

is used, where m_{ij} and σ_{ij} denote the mean and width of a fuzzy set A_{ij}, respectively. In the fuzzification process, crisp input x_j is converted into a fuzzy singleton and is mapped to the fuzzy set A_{ij} with degree $M_{ij}(x_j)$. In the inference engine, the fuzzy AND operation is implemented by the algebraic product in fuzzy theory. Given an input data set $\boldsymbol{x} = (x_1, ..., x_n)$, the firing strength $\mu_i(\boldsymbol{x})$ of rule i is calculated by

$$\mu_i(\boldsymbol{x}) = \prod_{j=1}^{n} M_{ij} = \exp\{-\sum_{j=1}^{n} (\frac{x_j - m_{ij}}{\sigma_{ij}})^2\} \tag{3}$$

The output from each rule is a crisp value. The fuzzy logic control action is the combination of the output of each rule using the weighted average defuzzification method. Suppose that a fuzzy controller consists of r rules, and then the output of the controller is

$$u = \frac{\sum_{i=1}^{r} \mu_i(\boldsymbol{x}) b_i}{\sum_{i=1}^{r} \mu_i(\boldsymbol{x})} \tag{4}$$

In applying HSEFC to the feedforward fuzzy controller design, only the number of rules should be assigned in advance. Instead of grid type partition, the rules are flexibly partitioned in the input space. If the total number of rules is set to r, then the number of fuzzy sets in each input variable as well as the number of fuzzy singletons in the consequent part are also equal to r.

2.2 Recurrent fuzzy controller

For a dynamic plant control, a recurrent controller appears to be a better choice than a feedforward controller. In the previous studies (Juang & Lin, 1999; Juang, 2002), performance recorded by applying recurrent fuzzy systems to dynamic problems solving has been shown to be superior to recurrent neural networks. The recurrent fuzzy controller designed in this chapter is a slight modification of that used in TSK-type recurrent fuzzy network (TRFN) (Juang, 2002) in that the consequent part is of zero-order instead of first-order TSK type. Figure 1 shows structure of the recurrent fuzzy system. Suppose the system consists of two rules. Each recurrent fuzzy if-then rule is in the following form

Rule i: IF $x_1(t)$ is A_{i1} AND … AND $x_n(t)$ is A_{in} AND $h_i(t)$ is G

THEN $u(t+1)$ is b_i AND $h_1(t+1)$ is w_{1i} AND $h_2(t+1)$ is w_{2i}, $i = 1, 2$ (5)

where A_{ij} and G are fuzzy sets, u is the output variable, h_i is the internal variable, w_{ij} and b_i are the consequent parameters for inference outputs h_i and u, respectively. The recurrent reasoning implies that the inference output $u(t+1)$ is affected by the internal variable $h_i(t)$, and the current internal output $h_i(t+1)$ is a function of previous output value $h_i(t)$, i.e., the internal variable $h_i(t)$ itself forms a recurrent reasoning. As in a feedforward fuzzy controller, Gaussian membership function is used for the fuzzy set A_{ij}. For the fuzzy set G, a global membership function $G(x) = 1/(1+e^{-x})$ is used. Given an input set $\boldsymbol{x} = (x_1, ..., x_n)$, the inference and internal outputs of the recurrent fuzzy controller are calculated, respectively, by

$$u(t+1) = \frac{\sum_{i=1}^{r} \mu_i(\boldsymbol{x}) b_i}{\sum_{i=1}^{r} \mu_i(\boldsymbol{x})} \tag{6}$$

and

$$h_i(t+1) = \sum_{k=1}^{r} \mu_k(\boldsymbol{x}) w_{ik} \tag{7}$$

where

$$\mu_i(\boldsymbol{x}) = G(h_i(t)) \cdot \prod_{j=1}^{n} M_{ij}(x_j) \tag{8}$$

In applying HSEFC to the recurrent fuzzy controller design, only the number of recurrent fuzzy rules should be assigned in advance. Suppose there are r rules in total, then the numbers of fuzzy sets A's on each external input variable x_i and the internal variable h_i, are all equal to r.

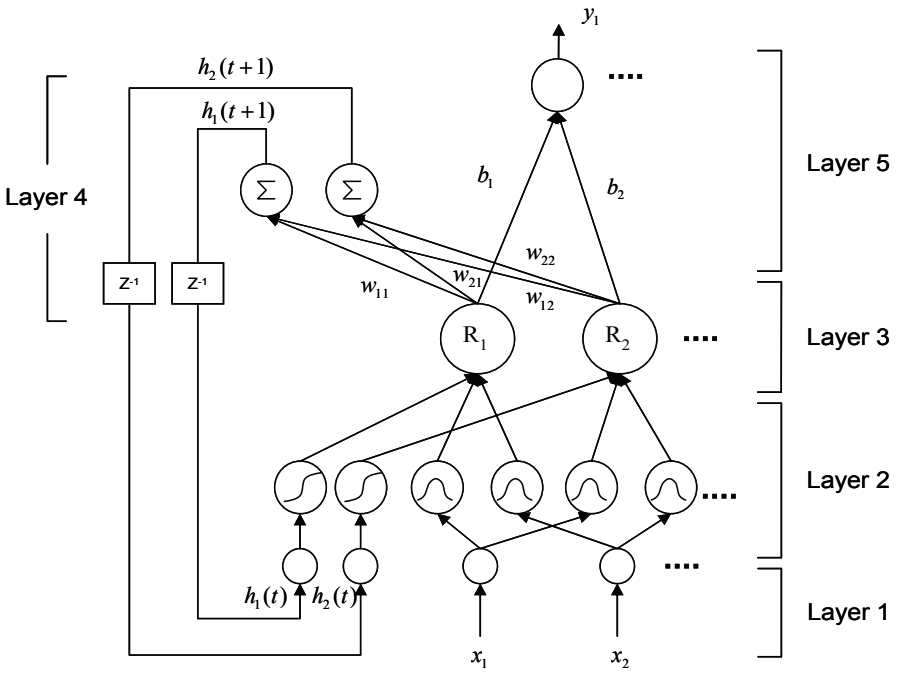

Fig. 1. Recurrent fuzzy system structure.

3. Symbiotic Evolution-based Fuzzy Controller (SEFC)

The symbiotic evolution-based fuzzy controller (SEFC) was proposed in (Juang et al., 2000), and the idea was used in many later studies (Mahfouf et al., 2001, Jamei et al., 2004, Kuo et al., 2004, Juang, 2005b, Lin & Xu, 2006). Unlike general GAs' evolution algorithms which operate on a population of full solutions to a problem (the Pittsburgh approach), symbiotic evolution assumes that each individual in the population represents only a partial solution; complete solutions are formed by combining several individuals. Figure 2(a) and (b) show

codings of a fuzzy system using the general and symbiotic evolutions, respectively. Each individual in Fig. 2(a) represents a whole fuzzy system. On the contrary, each individual in Fig. 2(b) represents a single fuzzy rule. In general GAs, a single individual is responsible for the overall performance, with the fitness value assigned to itself according to its performance. In symbiotic evolution, the fitness of an individual (a partial solution) depends on others. Partial solutions can be characterized as *specializations*. The specialization property tries to keep search diversity which prevents convergence of the population. The symbiotic evolution appears to be a faster and more efficient search scheme than the general evolution approaches for reinforcement learning problems (Moriarty & Miikkulainen, 1996; Juang et al., 2000; Lin & Xu, 2006).

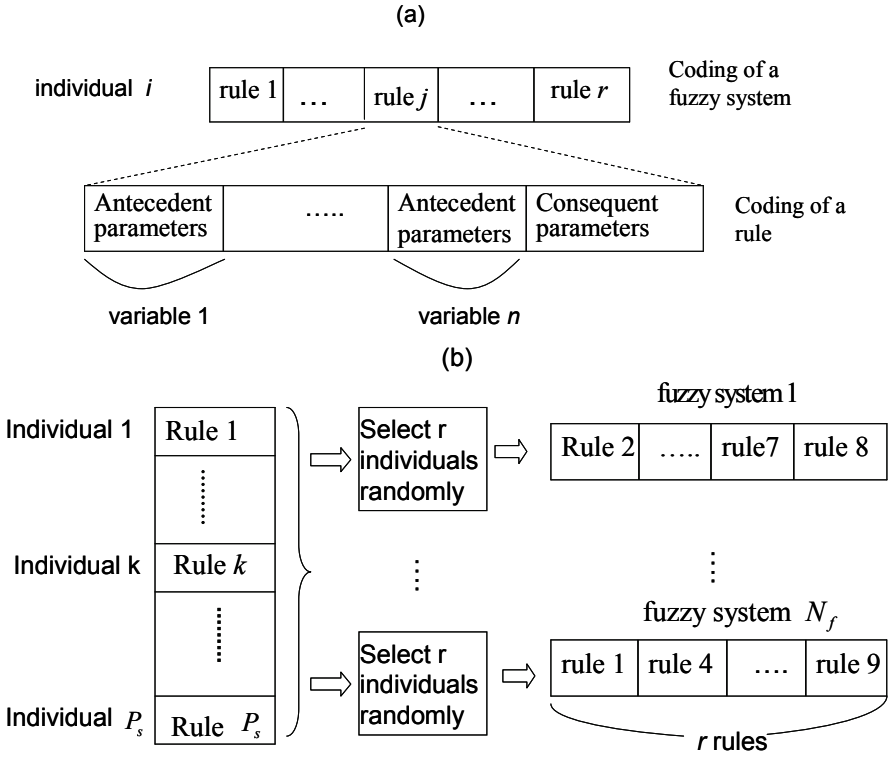

Fig. 2. Coding of a fuzzy system using (a) the general and (b) symbiotic evolutions.

The basic idea of SEFC is on the representation of a single fuzzy rule by an individual. A whole fuzzy system is formed by combining r randomly selected rules from a population. With the fitness assignment performed by symbiotic evolution and the local property of a fuzzy rule, symbiotic evolution and the fuzzy system design can complement each other. If a normal GA evolution scheme is adopted for fuzzy system design, only the overall performance of a fuzzy system is known, not the performance of each fuzzy rule. The method to replace the unsuitable fuzzy rules that degrade the overall performance of a fuzzy system is through random crossover operations, followed by observing the

performance of the offspring. Only when the overall performance of the fuzzy system is good do we know that the unsuitable rules have been replaced. In SEFC, the performance of each fuzzy rule may be implicitly evaluated. Such implicit evaluation is especially suitable for reinforcement learnimg problems which require only evaluation instead of instructive feedback information. With the local property of a fuzzy rule, the fitness assignment performed by the SEFC is quite representative. In this way, symbiotic evolution and fuzzy system design can complement each other, which result in a fast, efficient genetic search for reinforcement learning problems.

4. Hierarchical SEFC (HSEFC)

In SEFC, a GA search is performed only on the separate rules. The information about the participating rules in a well-performed fuzzy network is not propagated from generation to generation. If, besides the local rule search, we can propagate the information to the next generation and perform a global fuzzy network search simultaneously, then a better design performance could be achieved. This is the motivation of Hierarchical SEFC (HSEFC). This section introduces the detailed HSEFC implementation algorithm. Subsection 3.1 presents the HSEFC implementation algorithm for feedforward fuzzy controller design (HSEFC-F). In subsection 3.2, the HSEFC for recurrent fuzzy controller design (HSEFC-R) is presented.

4.1 HSEFC for feedforward fuzzy controller design (HSEFC-F)

Figure 3 shows the structure of the HSEFC-F design system. It consists of two populations. In population 1, each individual represents only a single fuzzy rule in the form described in (1). A whole fuzzy system constituted by r fuzzy rules is built by randomly selecting r individuals from population 1. The selected rules are recorded in an individual of population 2. Therefore, each individual in population 2 indicates a whole fuzzy system, with each gene representing a rule selected from population 1. Each constituted fuzzy controller in population 2 is applied to the plant to be controlled with a controller performance evaluation returned and used as the fitness value. This fitness value is assigned not only to the action system in population 2, but also distributed to the rules participating in the system. The concurrent evolution of populations 1 and 2 leads to an efficient algorithm. Detailed processes of these two stages are described as follows.

4.1.1 Local mapping stage

This stage performs evolution of population 1. Like general GAs, the evolution consists of three major operations: reproduction, crossover, and mutation. Initially, this stage randomly generates a population of individuals, each of which represents a set of parameters for the fuzzy rule in (1). The population size is denoted as P_{S1}, which indicates that P_{S1} fuzzy rules are generated. Each gene is represented by a floating number and the encoded form of each rule (individual) is as follows,

$$| m_{i1} | \sigma_{i1} | m_{i2} | \sigma_{i2} | ... | m_{in} | \sigma_{in} | b_i |$$

After creating a whole population with real-valued individuals, an interpreter takes one from the population and uses it to set part of the parameters in a fuzzy system. Suppose there are r rules in a fuzzy system, then a whole fuzzy system is constructed by selecting r

individuals from population 1. The fuzzy system runs in a feedforward fashion to control the plant until a failure or a success occurs. Then, in the reinforcement control problem, we should assign a credit to the fuzzy system. From the view point of temporal credit, if the fuzzy system can control the plant for a longer time, then the degree of success is higher and a higher credit should be assigned. Based on this concept, for each individual in population 2, the fitness value is assigned at the moment of failure.

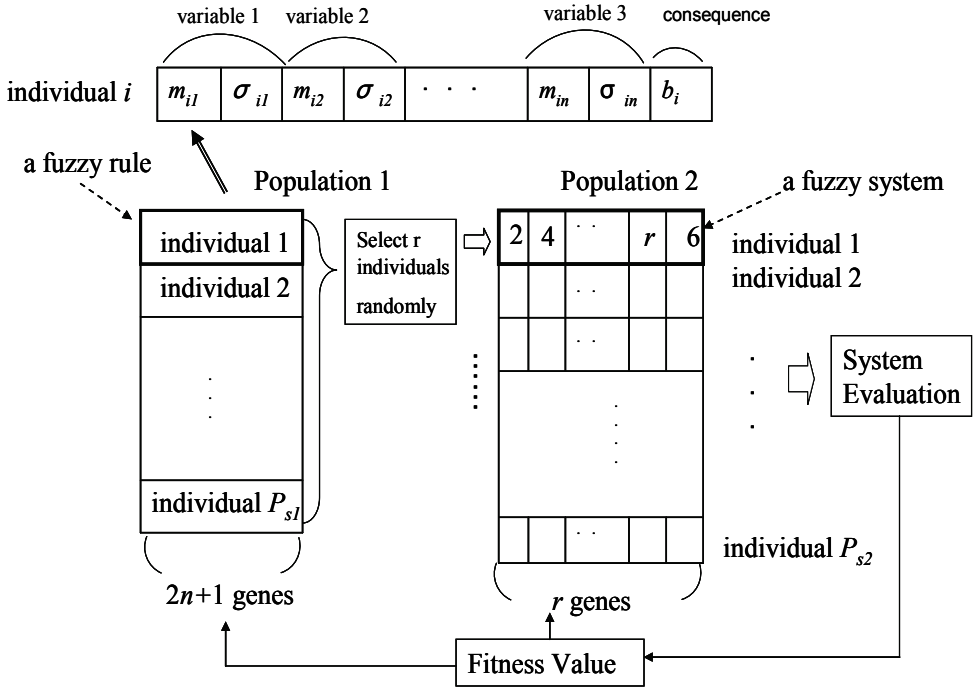

Fig. 3. Structure of the HSEFC for feedforward fuzzy system design (HSEFC-F).

To evaluate the performance of each individual in population 1, the credit assignment is not as direct as that used in population 2. We should apportion the credit to the participating rules in a fuzzy system. This credit assignment problem also occurs in the Michigan approach. In the Michigan approach, many different credit assignment schemes have been proposed (Cordón et al., 2001). The two most important ones are the bucket bridge algorithm (BBA) (Booker et al., 1989) and the profit sharing plan (PSP) (Grefenstette, 1988). The BBA adjusts the strength of an individual rule classifier by distributing the obtained reward across the sequence of active rule classifiers that are directly or indirectly contributed to the past actions by the fuzzy classifier system. It uses only the local interactions between rules to distribute credit. In contrast to the BBA, the PSP is a global learning scheme and typically achieves a better performance than the BBA. In (Ishibuchi et al., 1999), a simpler credit assignment algorithm is proposed. In this algorithm, there is always a unique winner rule to be rewarded or penalized depending on whether it correctly predicts the class of the training example. The fitness value of each rule is determined by the total reward assigned to the rule. Basically, the aforementioned schemes are based on an

economic analogy and consist of a bid competition between fuzzy rules. They measure only the goodness of an individual rule and do not consider the ability to cooperate with the remaining ones in the population. In fact, in the Michigan approach, since a population represents a whole system, each individual cooperates with the same ones in each generation. The quality of cooperation is difficult to obtain among these competing individuals.

In HSANE-F, many fuzzy systems are formed in each generation, and each individual may combine with different ones in each fuzzy system construction. By taking the average system fitness value in which an individual participates, we can approximately measure the individual cooperation ability. The measure is based on the fact that the fitness of each individual depends on the quality of the whole system it participates in, thus measuring how well it cooperates to solve the problem. As to the goodness of each individual, owing to the local mapping property, a well-performed rule will also have certain contribution to the system performance. On the contrary, a wrongly-mapped rule will degrade the system performance. The contribution of each rule to the system depends on its firing strength. However, the fitness value is available only when the control fails, during which the firing strength of each rule varies with time. It would be complex to distribute the fitness value among the participating rules based on the firing strength. A simple way is to equally distribute the system fitness value among the participating rules to measure its goodness. Therefore, by taking the average system fitness values in which an individual participates, we can approximately measure both the cooperation and goodness of the individual. Effectiveness of this fitness value distribution approach will be verified in simulations. Detailed steps of the approach are as follows.

Step 1. Divide the fitness value by r and accumulate the divided value to the fitness record of the r selected rules with their fitness set to zero initially.

Step 2. The above rule selection, plant control, and fitness division process are repeated P_{S2} (the size of population 2) times. The process ends when each rule has been selected for a sufficient number of times. Record the number of times each rule has participated in a fuzzy system.

Step 3. Divide the accumulated fitness value of each rule by the number of times it has been selected. The average fitness value represents the performance of an individual.

When the average fitness of each individual in population 1 is obtained, the HSEFC then looks for a better set of individuals to form a new population in the next generation by using genetic operators, including reproduction, crossover, and mutation. The detailed description of the three operations is as follows.

In reproduction operation, the elite strategy and tournament selection techniques are used. The top-half of best-performing individuals in the population are sorted according to their fitness value. Based on the elite strategy, these elites are advanced directly to the next generation. Also, to keep a non-decreasing best-so-far fitness value, the rules participating in the best-performing system in each generation are directly reproduced in the next generation. The remaining individuals are generated by performing tournament selection and crossover operations on the elites.

In crossover operation, the tournament selection scheme is used to select parents. Two individuals in the top-half of best-performing individuals are selected at random in the tournament selection, and their fitness values are compared. The individual with higher

fitness value is selected as one parent. The other is also selected in the same way. Two offspring are created by performing crossover on the selected parents. Here, one point crossover operation is performed. The top-half of worst-performing individuals are replaced by the newly produced offspring. The adopted crossover may be regarded as a kind of elite crossover.

In mutation operation, the gene in an individual is altered randomly by mutation. Uniform mutation is used, where a mutated gene is drawn randomly and uniformly from its search domain. In the following simulations, mutation probability p_m is set to 0.1.

The elite strategy above can improve the searching speed, but the population diversity may also be lost quickly at the same time. To overcome this disadvantage, a population renewal technique is used. In each generation, the relationship between each individual is monitored. Since half of the next generation population is generated by performing crossover on the top-half of best-performing individuals, it is only necessary to check the similarities between the top-half of best-performers. The cross correlation value between two neighbouring individuals in a performance ranked list is calculated and averaged. The mathematical form for this measure is as follows:

$$S = \frac{2}{P_{S1}} \sum_{i=1}^{P_{S1}/2-1} \frac{(D_i^T D_{i+1})^2}{D_i^T D_i D_{i+1}^T D_{i+1}} \tag{9}$$

where D_i is the i th best-performing individual in the rank list. The dimension of D_i is $\ell \times 1$, where ℓ is the number of genes in the individual. With this measure, if all of the individuals are exactly the same, then S is equal to 1. This similarity measure is performed for each generation. When the measurement similarity is higher than a pre-specified threshold T_{hr}, it reflects that the elites have moved to a degree of convergence. If this phenomenon occurs, then most parts of the individuals are renewed. In the renewal process, only a portion of the top best-performing individuals are reproduced to the next generation, and the remaining parts are replaced by newly generated individuals. After the renewal process, the similarity value is again calculated on each subsequent generation, and the renewal process is performed when the value is higher than the threshold T_{hr}. The renewal process can always keep the diversity of the population and thus helps to find the best fuzzy rules.

4.1.2 Global mapping stage

This stage performs evolution of population 2. The function of population 2 consists of both exploitation and exploration of the rule-combination in a fuzzy system. In exploitation, the information about which rules are combined together in a well-performed fuzzy system is propagated from generation to generation. On the other hand, evolutionary procedure is performed on population 2 to search the best-combination in exploration. Without population 2, in each generation, the rules participating in a fuzzy system should be randomly selected from population 1. Population 2 helps to concentrate the search on the best rule combination. Since populations 1 and 2 are evolved concurrently, if individuals in the former are updated frequently, the search in the latter might be meaningless. To avoid this phenomenon, as stated in the local-mapping-search-stage, the top-half of best-performing individuals in population 1 are reproduced directly to the next generation. Only

the remaining poorly-performed individuals are updated. Owing to the local mapping property, the update of these rules has only local influence on its participating fuzzy system. In general, the newly generated rules outperform the original poorly-performing ones. So the evolution of population 1 is also helpful to that of population 2, that is, both are cooperative. This property will be verified in the simulations.

In this stage, an integer-value encoding scheme is used, and the alleles have values in the set {1, 2, …, P_{S1}}. There are r genes in each individual, and it has the following form,

$$| 2 | 7 | 5 | 9 | \ \dots\ | P_{S1} | \ \dots\ | 8 | 1 |$$

The population size of population 2 is set to be P_{S2}, indicating that P_{S2} fuzzy controllers are applied to plant control in each generation. Due to the following two reasons, the genetic operation used in this stage is different from that used in the local-mapping search stage. First, since a flexible partition of precondition part is adopted and reinforcement learning is performed, the rule number r in a fuzzy system is usually small. Population 2 converges quickly due to the small individual length. Second, the character of population 2 in the whole searching task is auxiliary and should always maintain diversity to coordinate the evolution of population 1 from generation to generation. If population 2 converges faster than population 1, then the evolution of population 1 is useless. In order to maintain population diversity, the following genetic operation is used. The top-half of best-performing individuals in population 2 are sorted according to their fitness values. To select the parents for crossover operation, the tournament select scheme is adopted and performed on the top-half of best-performing individuals. By performing the one-point crossover operation on the selected parents, offspring can be created and half of the new population is produced in this way. As stated in the local-rule searching stage, it is desired to maintain a non-decreasing best-so-far fitness value, consequently the best-performing individual is directly reproduced in the next generation. As to the remaining half of the population, they are created by randomly generated individuals. For the mutation operation, a mutated gene is selected randomly and uniformly from the integer set {1, 2, …, P_{S1}}. The mutation probability is set to 0.1.

After the above crossover and mutation operations, overlapping of rules might occur. If this occurs, then the total number of rules in a fuzzy system is less than r. For this situation, the overlapping of each rule is regarded as a weighting factor F of its firing strength. If the overlapping number of rule i is n_i, then $F_i = n_i$. With the weighting factor, the output equation of the fuzzy system in (4) can be rewritten as

$$u = \frac{\sum_{i=1}^{r'} \mu_i(\boldsymbol{x}) F_i b_i}{\sum_{i=1}^{r'} \mu_i(\boldsymbol{x}) F_i} \tag{10}$$

where $r' \le r$ is the total number of rules and $\sum_{i=1}^{r'} F_i = r$.

4.2 HSEFC for recurrent fuzzy controller design (HSEFC-R)

This subsection introduces the application of HSEFC to recurrent fuzzy controller design. The recurrent fuzzy rule to be designed is previously described in (5). By regarding each recurrent rule as an individual in population 1 of HSEFC-F, the recurrent fuzzy controller can be designed. However, this approach does not use the recurrent fuzzy system structure to its full advantage. To speed up the design process, the HSEFC-R designed specifically for the recurrent fuzzy rule is proposed. The divide-and-conquer concept is incorporated in HSEFC-R (Juang, 2005b). Based on this concept, the recurrent fuzzy rule in (5) is decomposed into two sub-rules, the spatial sub-rule and the temporal sub-rule. The antecedent parts of both sub-rules are the same as that in (5). The consequent part in each spatial sub-rule considers only the output variable u, while the consequent part in each temporal sub-rule considers only the variables h_1, …, h_r. In HSEFC-R, the spatial and temporal sub-rules are evolved simultaneously. Figure 4 shows the HSEFC-R structure,

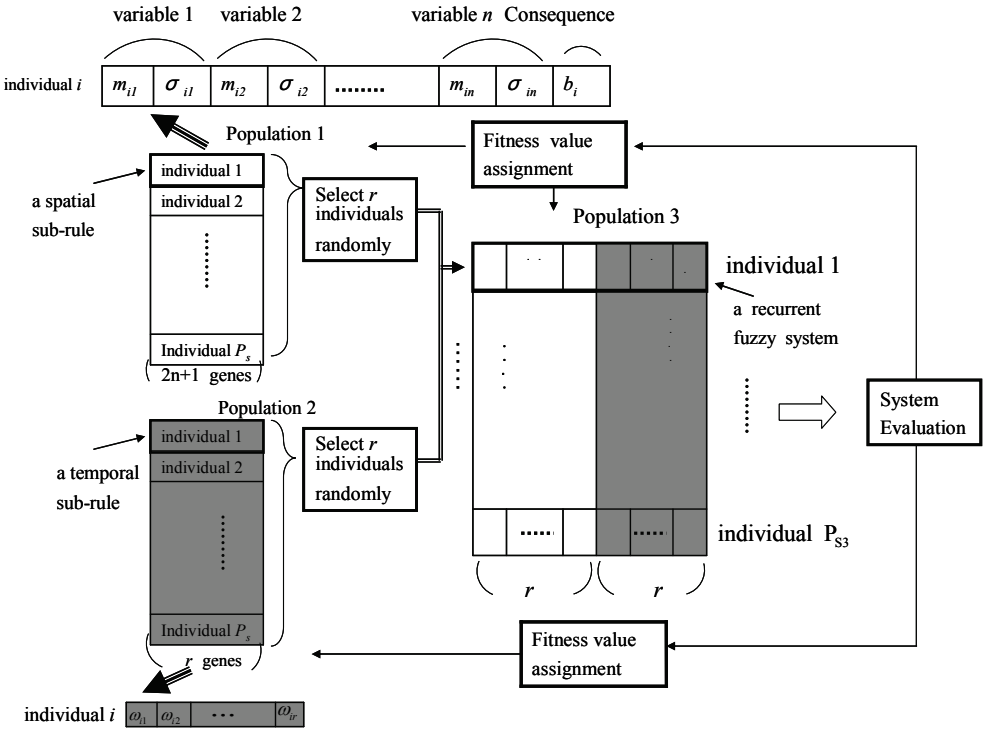

Fig. 4. Structure of the HSEFC for recurrent fuzzy system design (HSEFRC-R).

where there are three populations. Populations 1 and 2 are responsible for spatial and temporal sub-rules searches, respectively. Population 3 is responsible for the whole recurrent fuzzy system search. Each individual in population 1 represents a spatial sub-rule, whereas each individual in population 2 represents a temporal sub-rule. Since the spatial and temporal sub-rules share the same antecedent part, the antecedent parameters are encoded in population 1 only. A recurrent fuzzy system consisting of r rules is constructed

by randomly selecting r individuals from both populations 1 and 2. The selected individuals from both populations are recorded in population 3. Each individual in population 3 represents a whole fuzzy system. The task of creating population 3 is not only to search the best combinations of the r spatial sub-rules or temporal sub-rules selected from each population, but also to search for the best match of both types of sub-rules. Each recurrent fuzzy system encoded in population 3 is applied to a dynamic plant control with the return of a performed evaluation. The evaluation is used as the fitness value of the controller. As in HSEFC-F, the fitness value for each individual in population 3 is set to the time steps until failure for each control trial. This fitness value is then assigned to the participating sub-rules selected from populations 1 and 2. With the distributed fitness value, evolution of populations 1 and 2 is performed in the local-mapping search stage, while evolution of population 3 is performed in the global-mapping search stage. These two stages are executed concurrently until a successful control is achieved. Detailed operations of these two stages are described as follows.

4.2.1 Local mapping stage

The objective of this stage is to explore the well-performing spatial and temporal sub-rules in each local input region. First, populations 1 and 2 are created by randomly generated individuals. The sizes of both the populations are equal to and are denoted as P_S. The real-value encoding scheme is used in both populations. Each individual in population 1 encodes a spatial sub-rule and has the following form:

$$| m_{i1} | \sigma_{i1} | m_{i2} | \sigma_{i2} | ... | m_{in} | \sigma_{in} | b_i |$$

Each individual in population 2 encodes only the consequent part of a temporal sub-rule because the spatial and temporal sub-rules share the same antecedent part. Each individual has the following form:

$$| w_{i1} | w_{i2} | ... | w_{ir} |$$

The relationship between the individuals in populations 1 and 2 is cooperative. The genetic operation of each population is executed independently and concurrently. The fitness value decision method of an individual is similar to that used in HSEFC-F. If the fitness value of the recurrent fuzzy system consisting of r recurrent rules is *Fit*, then the distributed fitness value of each participating individuals from populations 1 and 2 is set to *Fit/r*. When the fitness value of each individual in both populations is given, new populations are generated by using genetic operations. Like HSEFC-F, the elite strategy and tournament selection techniques are used here. The reproduction, crossover, and mutation operations are the same as those used in the local-mapping search stage of HSEFC-F. The mutation probability is set at 0.1. To keep population diversity, the population renewal technique is applied to both the populations. A threshold value, T_{hr}, is set for both populations. If the similarity value of the top-half of best-performing individuals is higher than T_{hr} in each individual, then the renewing technique is applied to that population.

4.2.2 Global mapping stage

This stage performs evolution of population 3. An integer-value encoding scheme is used. Each individual contains $2r$ genes. The first r genes represent the r spatial sub-rules

selected from population 1, while the remaining r genes represent the r temporal sub-rules selected from population 2. Each gene representing the selected sub-rule has value in the integer set {1, 2, …, P_S }. Each individual has the following form

$$|7|2|4|\ldots|P_S|\ \ldots|11|4||2|13|7|\ldots|P_S|\ldots|15|4|$$

The temporal sub-rule recorded in position $r+k$ shares the same antecedent part with the spatial sub-rule in position k . The population size is set to P_{S3} , indicating that P_{S3} recurrent fuzzy controllers are built and applied to a dynamic plant control in each generation. The fitness value of each individual is assigned according to the controller performance evaluation. For the genetic operation, in addition to the crossover operation, the other operations used are the same as those used in the global-mapping search stage of HSEFC-F. In the crossover operation, to exchange the spatial and temporal sub-rule combination information of each population, a two-point crossover operation is performed. One crossover site is located at the first r genes, indicating the exchange of spatial sub-rule combination information; the other is located at the last r genes, indicating the exchange of temporal sub-rule combination information.

5. Simulations

This section presents simulation results of HSEFC for feedforward and recurrent fuzzy controller design under genetic reinforcement learning environments. All simulations in the following examples are written in C++ program, and run on a Pentium-1G personal computer. For the fuzzy rule number selection, it is somewhat heuristic and depends on the complexity of the plant to be controlled. In the following examples, the number of rules in each fuzzy system is set to five, i.e., r = 5.

5.1 Feedforward fuzzy controller design

Example 1. Cart-Pole Balancing System. In this example, HSEFC-F is applied to a classic control problem referred to as the cart-pole balancing problem. This problem is often used as an example of inherently unstable and dynamic systems to demonstrate both modern and classic control techniques, and is now used as a control benchmark (Andersonm 1989). The cart-pole balancing problem is the problem of learning how to balance an upright pole. 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/second); x, the horizontal position of the center of the cart (in meters); and $\dot{x}$, the velocity of the cart (in m/s). The only control action is u, which is the amount of force (Newton) applied to the cart to move it toward its left or right. The system fails when the pole falls past a certain angle (12 degrees is used here) or the cart runs into the bounds of its track (the distance is 2.4m from the center to both bounds of the track). Details of the control system description can be found in (Juang et al. 2000). A control strategy is deemed successful if it can balance a pole for 120000 time steps. In designing the fuzzy controller, the four states are fed as the controller inputs, and the controller output is u . In HSEFC-F, the number of individuals (rules) in population 1 is set to 50 (i.e., P_{S1} =50). The size of population 2 is set to 50 (i.e. P_{S2} =50), indicating that fifty

fuzzy controllers are built and evaluated per generation. The evaluation of a fuzzy controller consists of a single trial to the cart-pole system. The similarity measure threshold T_{hr} in the renewing process is set at 0.5. The fitness value is equal to the number of time steps in which the pole remains balanced. For each control trial, the initial values of $(x, \dot{x}, \theta, \dot{\theta})$ are random values in region [-2, 2]x[-1.5, 1.5]x[-5, 5]x[-40, 40]. In this example, 100 runs are simulated, and a run ends when a successful controller is found or a failure run occurs. The definition of a failure run is if no successful fuzzy controller is found after 25,000 trials. The number of pole-balance trials and the CPU time (the time from the first trial to the end of a successful trial) are measured. The average CPU time and trial number of the HSEFC-F are 4.0 (sec) and 179, respectively. Figure 5 (a) and (b) show the control results of position and angle in the first 1000 time steps of three different runs with different initial states. For SEFC, the average results are 5.1 (sec) and 256 trials. The results show that the performance of HSEFC-F is better than SEFC. Since the performance of SEFC has been shown to be better than other compared reinforcement learning methods in (Juang et al., 2000), only SEFC is compared this example.

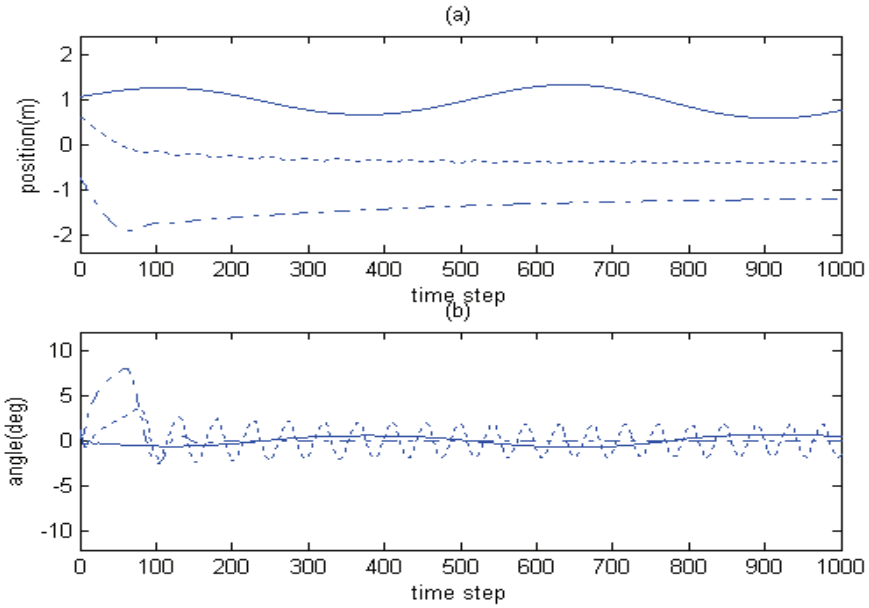

Fig. 5. Control results of (a) position (b) angle in the first 1000 time steps of three different runs with different initial states in Example 1.

5.2 Recurrent fuzzy controller design

Example 2. Dynamic Plant Control. The dynamic plant to be controlled is described by the following equation

$$y_p(k+1) = 0.6y_p(k) + 0.03y_p(k-1)u(k) + 0.01u^2(k-2) + 0.2u(k-3) \tag{11}$$

The current output of the plant depends on two previous outputs and four previous inputs. In (Kim et al., 1998), it is shown that for this type of plant, a poor performance is achieved by a linear predictive control. The controller input u is in the range [-20, 20]. The initial states are $y_p(0) = y_p(1) = 3$. The regulation point y_{ref} is set to 10. The performance of a controller is measured by the number of time steps in which the controlled state y_p satisfies the following constraint. The constraint set starts from the initial state and after 10 times of control, the state of y_p should be within the region [y_{ref} -0.2, y_{ref} +0.2], otherwise a failure occurs. A recurrent fuzzy controller designed by HSEFC-R is applied to the plant. In HSEFC-R, the sizes of populations 1, 2, and 3 are all set to 100. The similarity measure threshold T_{hr} in the renewal process is set to 0.35. Since a recurrent fuzzy controller is used, only the current state $y_p(k)$ and reference state y_{ref} are fed as the controller inputs. Since a recurrent fuzzy controller consists of five recurrent fuzzy rules, the number of genes in each individual of populations 1 or 2 is equal to 5. One hundred runs are simulated, and a run ends when a successful controller is found. A failure run is said to occur if no successful fuzzy controller is found after 100,000 trials. The average CUP time and trial number of HSANE-R are 0.65 (sec) and 1853, respectively. For SEFC, the results are 1.37 (sec) and 3960 trials. The performance of HSANE-R is much better than SEFC. Detailed comparisons of different design methods can be found in (Juang, 2005b).

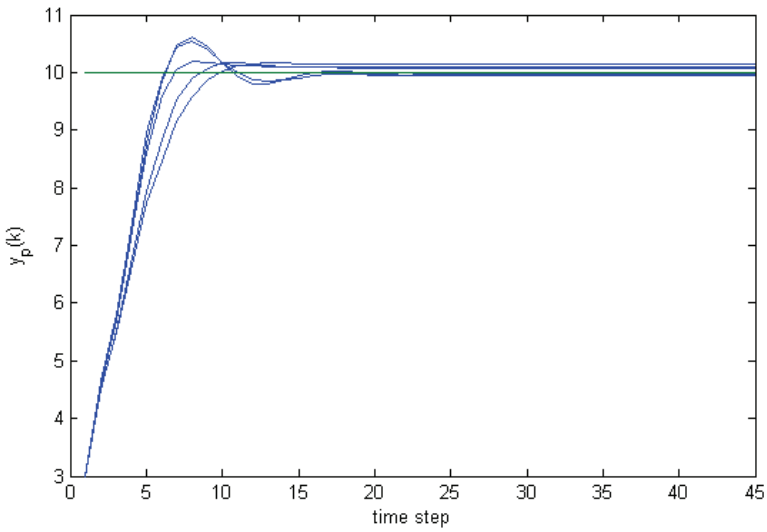

Fig. 6. Dynamic plant control results of five different runs using HSEFC-R in Example 2.

6. Conclusion

This chapter introduces a unified symbiotic evolution framework (the HSEFC) for feedforward and recurrent fuzzy controller design in reinforced learning environments. The

design of a fuzzy controller is divided by the HSEFC into two iterative search stages: the local-mapping search stage and the global-mapping search stage. In this way, the population in each stage is evolved independently and concurrently. Furthermore, to avoid the premature population phenomenon, modifications of general genetic operations are also incorporated in the design process. In the interests of utility and economy, the HSEFC operates under two formats; HSEFC-F and HSEFC-R. For feedforward fuzzy controller design, HSEFC-F is presented, while for recurrent fuzzy controller design, HSEFC-R, which uses the divide-and-conquer technique on spatial and temporal sub-rules search, is presented. As shown, simulation results in static and dynamic plant control problems have verified the effectiveness and efficiency of HSEFC-F and HSEFC-R. Although in HSEFC solutions, the designed fuzzy controller structure is currently assigned in advance, further work on HSEFC intends to focus on its extension to automatic controller structure determination. A more accurate fitness assignment for each single rule in symbiotic evolution is another future research topic.

7. References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, Vol. 9, 31-37.

Belarbi, K. & Titel, F. (2000). Genetic algorithm for the design of a class of fuzzy controllers: an alternative approach. *IEEE Trans. Fuzzy Systems*, Vol. 8, No. 4, 398-405.

Bonarini, A. (1993). ELF: learning incomplete fuzzy rule sets for an autonomous robot, *Proc. 1st European Congress on Intelligent Technologies and Soft Computing*, pp. 69-75, Aachen, Germany.

Booker, L.B.; Goldberg, D.E. & Holland, J.H. (1989). Classifier systems and genetic algorithms. *Artificial Intelligence*, Vol. 40, 235-282.

Chou, C.H. (2006). Genetic algorithm-based optimal fuzzy controller design in the linguistic space. *IEEE Trans. Fuzzy Systems*, Vol. 14, No. 3, 372-385.

Chung, IF.; Lin, C.J. & Lin, C.T. (2000). A GA-based fuzzy adaptive learning control network. *Fuzzy Sets and Systems*, Vol. 112, No. 1, 65-84.

Cordón, O.; Herrera, F.; Hoffmann, F. & Magdalena, L. (2001). *Genetic Fuzzy Systems: Evolutionary Tuning and Learning of Fuzzy Knowledge Bases*, World Scientific.

Cordón, O.; Gomide, F.; Herrera, F.; Hoffmann, F. & Magdalena, L. (2004). Ten years of genetic fuzzy systems: current framework and new trends. *Fuzzy Sets and Systems*, Vol. 141, No. 1, 5-31.

Goldberg, D.E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley.

Grefenstette, J.J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Leaqrning*, Vol. 8, 225-246.

Furuhashi, T.; Nakaoka, K. & Uchikawa, Y. (1995). An efficient finding of fuzzy rules using a new approach to genetic based machine learning, *Proc. 4th IEEE Int. Conf. Fuzzy Systems*, pp. 715-722, Yokohama, Japan.

Homaifar, A. & McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Trans. Fuzzy Systems*, Vol. 3, No. 2, 129-139.

Ishibuchi, H.; Nakashima, T. & Murata, T. (1999) Performance evaluation of fuzzy classfier systems for multidimentioinal pattern classification problems. *IEEE Trans. On Systems, Man and Cybernetics, Part B: Cybernetics*, Vol. 29, 601-618.

Jamei, M.; Mahfouf, M. & Linkens, D.A. (2004). Elicitation and fine-tuning of fuzzy control rules using symbiotic evolution. *Fuzzy Sets and Systems*, Vol. 147, No. 1, 57-74.

Juang, C.F. & Lin, C.T. (1999). A recurrent self-organizing neural fuzzy inference network. *IEEE Trans. Neural Networks*, Vol. 10, No. 4, 828-845.

Juang, C.F.; Lin, J.Y. & Lin, C.T. (2000). Genetic reinforcement learning through symbiotic evolution for fuzzy controller design. *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, Vol. 30, No. 2, 290-302.

Juang, C.F. (2002). A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms. *IEEE Trans. Fuzzy Systems,* Vol. 10, No. 2, 155-170.

Juang, C.F. (2004). Temporal problems solved by dynamic fuzzy network based on genetic algorithm with variable-length chromosomes. *Fuzzy Sets and Systems*, Vol. 142, No. 2, 199-219.

Juang, C.F. (2005a). Combination of on-line clustering and Q-value based GA for reinforcement fuzzy system design. *IEEE Trans. Fuzzy Systems*, Vo. 13, No. 3, 289-302.

Juang, C.F. (2005b). Genetic recurrent fuzzy system by coevolutionary computation with divide-and-conquer technique. *IEEE Trans. Systems, Man, and Cybernetics, Part C: Applications and Reviews* , Vol. 35, No. 2, 249-254.

Karr, C.L. (1991). Design of an adaptive fuzzy logic controller using a genetic algorithm, *Proc. the Fourth Int. Conf. Genetic Algorithms*, pp. 450-457.

Kim, J.H.; College, D.T.; Gun, A. & DO, G. (1998). Fuzzy model based predictive control, *Proc. IEEE. Int. Conf. Fuzzy Systems*, pp. 405-409, Anchorage, AK, USA.

Kuo, H.C.; Chang, H.K. & Wang, Y.Z. (2004). Symbiotic evolution-based design of fuzzy-neural diagnostic system for common acute abdominal pain. *Expert Systems with Applications*, Vol. 27, No. 3, 391-401.

Lee, C.H. & Teng, C.C. (2000). Identification and control of dynamic systems using recurrent fuzzy neural networks. *IEEE Trans. Fuzzy Systems*, Vol. 8, No. 4, 349-366.

Lin, C.T. & Jou, C.P. (1999). Controlling chaos by GA-based reinforcement learning neural network. *IEEE Trans. Neural Networks*, Vol. 10, No. 4, 846-859.

Lin, C.T. & Lee, C.S.G. (1996). *Neural Fuzzy Systems: A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice Hall, USA.

Lin, C.J. & Xu, Y.J. (2006). A self-adaptive neural fuzzy network with group-based symbiotic evolution and its prediction applications. *Fuzzy Sets and Systems*, Vol. 157, Issue 8, 1036-1056.

Mahfouf, M.; Jamei, M. & Linkens, D.A. (2001). Rule-base generation via symbiotic evolution for a Mamdani-type fuzzy control system, *Proc. the 10th IEEE Int. Conf. Fuzzy Systems*, pp. 396–399.

Miller, W.T.; Sutton, R.S. & Werbos, P.J. (1990). *Neural Networks for Control*, The MIT Press.

Moriarty, D.E. & Miikkulainen, R. (1996). Efficient reinforcement learning through symbiotic evolution. *Machine Learning*, Vol. 22, 11-32.

Moriarity, D.E. & Miikkulainen, R. (1998). Hierarchical evolution of neural networks, *Proc. IEEE Conf. Evoulutionary Computation*, pp. 428-433, Anchorage, AK, USA.

Paredis, J. (1995). Coevolutionary computation. *Aftifical Life*, Vol. 2, No. 4, 355-375.

Peña -Reyes, C.A. & Sipper, M. (2001). Fuzzy CoCo: a cooperative-coevolutionary approach to fuzzy modelling. *IEEE Trans. Fuzzy Systems*, Vol. 9, No. 5, 727-737.

Potter, M.A.; Jong, K.D. & Grefenstette, J. (1995). A coevolutionary approach to learning sequential decision rules, *Proc. 6th Int. Conf. Genetic Algorithms*, pp. 366-372, Pittsburgh, USA.

Potter, M.A. & DeJong, K.A. (2000). Cooperative coevolution: An architectural for evolving coadopted subcomponents. *Evol. Computation*, Vol. 8, No. 1, 1-29.

Shi, Y.; Eberhart, R. & Chen, Y. (1999). Implementation of evolutionary fuzzy systems. *IEEE Trans. Fuzzy Systems*, Vol. 7, No. 2, 109-119.

Sutton, R.S. & Barto, A.G. (1998). *Reinforcement Learning*, The MIT Press.

Whitley, D.; Dominic, S.; Das, R. & Anderson, C.W. (1993). Genetic reinforcement learning for neurocontrol problems. *Machine Learning*, Vol. 13, 259-284.

Valenzuela-Rendon, M. (1991). The fuzzy classifier system: A classifier system for continuously varying variables, *Proc. 4th Int. Conf. Genetic Algorithms*, pp. 346-353, San Diego, USA.

Zhao, Q. (1998). A general framework for cooperative co-evolutionary algorithms: a society model, *Proc. IEEE Conf. Evolutionary Computation*, pp. 57-62, Anchorage, AK, USA.

Zhang, J. & Morris, A.J. (1999). Recurrent neuro-fuzzy networks for nonlinear process modeling. *IEEE Trans. Neural Networks*, Vol. 10, No. 2, 313-326.

Advances in Evolutionary Algorithms

Edited by Xiong Zhihui

ISBN 978-953-7619-11-4
Hard cover, 284 pages
Publisher InTech
Published online 01, November, 2008
Published in print edition November, 2008

With the recent trends towards massive data sets and significant computational power, combined with evolutionary algorithmic advances evolutionary computation is becoming much more relevant to practice. Aim of the book is to present recent improvements, innovative ideas and concepts in a part of a huge EA field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Chia-Feng Juang and I-Fang Chung (2008). Symbiotic Evolution Genetic Algorithms for Reinforcement Fuzzy Systems Design, Advances in Evolutionary Algorithms, Xiong Zhihui (Ed.), ISBN: 978-953-7619-11-4, InTech, Available from: http://www.intechopen.com/books/advances_in_evolutionary_algorithms/symbiotic_evolution_genetic_algorithms_for_reinforcement_fuzzy_systems_design

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821