Fuzzy Modeling Employing Fuzzy Polyploidy Genetic Algorithms

MING-DA WU AND CHUEN-TSAI SUN
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: ray@cis.nctu.edu.tw

Fuzzy modeling generally comprises structure identification and parameter identification. The former determines the structure of a rule-base, whereas the latter determines the contents of each rule. Applying neural networks or genetic algorithms to identify the parameter sets and structures of a fuzzy system is increasingly popular owing to their ability to learn and adapt. However, most conventional approaches cannot integrate structure identification and parameter identification efficiently. This work presents a general approach to fuzzy modeling, i.e, fuzzy polyploidy genetic algorithms which integrate structure identification and parameter identification in a single evolution process. Capable of simulating the structural adaptation process of natural evolution, the proposed model is a generalized model for simultaneously optimizing both structure and parameters of fuzzy rule-bases. The structural adaptation proposed herein provides complete structural operations to simulate structural variation process and simple to complex life form of natural evolution. Illustrative examples involving typical FLCs, such as Mamdani and TSK models, demonstrate the effectiveness of applying the polyploidy scheme.

Keywords: fuzzy modeling, genetic algorithms, fuzzy logic controllers, polyploidy, structural adaptation

1. INTRODUCTION

Fuzzy concept [1] is widely used in rule representation owing to its ability to provide a convenient and intuitive means of describing the uncertainty of knowledge. However, modeling a fuzzy system is not easy in some applications due to the difficulty of extracting knowledge from experts. Therefore, learning techniques such as neural networks (NNs) [2, 3] and evolutionary computing (EC) [4] have been applied to model a fuzzy system by learning from numerical data. Many approaches have been developed to integrate fuzzy concept, NN, and ECs to construct intelligent systems. These approaches form the framework of current soft computing [5, 6]. Integration of fuzzy, NN, and EC is also referred to as computational intelligence [7, 8] because these approaches also construct intelligent systems by simulating natural biological mechanisms. A review of the literature on current trends in soft computing can be found in [9]. Integrating these techniques provides an appropriate way for fuzzy modeling.

Received December 21, 2000; revised July 11, 2001; accepted August 22, 2001.
Communicated by Wen-Lian Hsu.

Integration of fuzzy logic and neural networks can be categorized into either fuzzy neural networks [10, 11] or neural fuzzy systems [12]. Fuzzy neural networks contains inputs, outputs and weights between nodes that are represented in fuzzy forms, whereas neural fuzzy systems contains fuzzy rules or knowledge that are represented in a neural network format. Fuzzy neural networks concentrate on enhancing the convergence, flexibility and adaptability of the conventional neural network models by applying fuzzy theory, an example of which can be found in [13]. In contrast, neural fuzzy systems optimize fuzzy systems with the assistance of neural network adaptability, and has been extensively applied in designing fuzzy adaptive controllers [14]. However, although employing neural fuzzy systems is a feasible approach for fuzzy modeling, the learning of neural networks has the potential risk of finding a local optimum instead of the global optimum. Thus the initial weights and learning rate must be carefully controlled. Also, different aspects of the network structure, such as the number of nodes, are difficult to determine. Moreover, although most neural fuzzy systems rely on supervised learning to adjust the weights between neurons, the training data used for supervised learning in certain applications is difficult to obtain. Hence, those applications require reinforcement learning such as evolutionary computation [15].

In addition to neural networks, evolutionary computation has been studied for fuzzy modeling. Typical evolutionary computation, which simulates natural evolutionary processes to optimize the probability of survival in a changing environment, is an effective and efficient means of solving optimization, search, and learning problems. The study of evolutionary computation can be categorized into evolutionary programming [16], genetic algorithms [17, 18], evolutionary strategies [19, 20], genetic programming [21-23] and classifier systems [24]. In addition to utilizing a population and genetic operation such as mutation, all of these strategies optimize the population by simulating natural selection. Evolutionary programming (EP) attempts to evolve intelligence behavior by using the chromosome of a finite state machine (FSM) as a candidate solution. Holland [17, 18] proposed an even more general optimization model, genetic algorithms. Genetic algorithms introduce the crossover operation in addition to the mutation operation of EP, and have become the conventional method of EC in solving real-world optimization problems. The evolutionary strategy (ES) developed by Rechenberg and Schwefel is an effective means of solving parameter optimization problems. Holland's classifier, which can be treated as a special application of GAs, focuses on constructing a rule-base by evolution. A literature survey relating to evolution computation can be found in [25, 26], and related textbooks [4, 27, 28].

The feasibility of integrating evolutionary computation and fuzzy systems, commonly known as *evolutionary fuzzy systems* [29], has received considerable interest in recent years. Since most evolutionary fuzzy systems employ genetic algorithms, they are also referred to as *genetic fuzzy systems* in some studies [30]. A related survey can be found in [9, 31]. Early studies (1990-1995) used EC to optimize fuzzy membership, fuzzy rules, or both. Karr attempted to determine the fuzzy membership of a fuzzy controller by using genetic algorithms [32, 33]; this approach also was tested in [34-36]. Thrift used a genetic algorithm to find fuzzy rules of a fuzzy logic controller in which fuzzy membership is pre-defined [37]. Different approaches for this direction were proposed in [32-39]. Lee and Takagi proposed an evolutionary approach to simultaneously derive fuzzy memberships and rule sets of fuzzy controllers [40]. A survey of related research can also be found in [41-43]. In this work we present a general encoding framework, fuzzy polyploidy, for fuzzy rule set representation (including rule antecedence, consequence, and fuzzy membership). In doing so, fuzzy memberships and rule sets can be simultaneously optimized via evolution. Also proposed here is a novel, simple to complex process accompanying fuzzy polyploidy, in order to maintain the efficiency of GAs in searching a large space with complex structure.

Recent developments in evolutionary fuzzy systems has moved in several distinct, but not conflicting, directions. First, evolutionary fuzzy systems have been applied to real world problems. Kiguchi used [44] genetic algorithms in fuzzy-neuro controllers to control a robot in an unknown environment. Second, engineering methods have been employed to improve their performance, such as reducing the search time or memory usage. Streifel et al. [45] used dynamic parameter encoding to increase the speed of convergence and the accuracy of a genetic fuzzy controller. Third, hybrid learning approaches have been developed to enhance the learning ability of evolutionary fuzzy systems. Pedrycz [46] combined GAs and gradient-based techniques to solve fuzzy relational equations. Herrera et al. [47] proposed a three-stage learning process: a fuzzy rule genetic generating process, a rule number optimization process and a fuzzy membership adjusting process. Cordon [48] proposed a hybrid evolutionary process composed of generic algorithms and evolutionary strategies for rule refinement and rule generation. Literature surveys in this direction can be found in [49, 50]. Fourth, novel techniques from evolutionary algorithms have been introduced to improve the evolutionary fuzzy system. Cheong and Lai [51] proposed a parallel GA approach to concurrently evolve three populations with 3×3, 5×5 and 7×7 Mamdani FLC respectively. Thus, an adequate rule-base size can be obtained via evolution. However, their approach is limited by the lack of the efficiency in chromosome encoding, thus making it difficult to extend to modeling a complex FLC. Juang et al. [52] introduced a symbiotic evolution concept [53] in designing fuzzy controllers, thereby providing the advantages of reducing learning time. However, their approach cannot automatically determine the rule-base size. The fuzzy polyploidy approaches proposed here, which provide a novel, general scheme for modeling fuzzy systems, belongs to the fourth direction. The polyploidy encoding scheme as observed from nature provides a flexible, dynamic chromosome length encoding scheme for representing rule sets. In addition, the structural adaptation facilitates automatic determination of an adequate rule-base size via evolution.

In addition to common genetic algorithms, Holland's classifier system [54], which was designed as a general method to represent and adapt a rule-base, has been used to adapt a fuzzy rule-base. Classifier systems can be roughly divided into two categories, i.e., the Michigan [54] and Pittsburgh [55] approaches. The Michigan approach treats the entire population as a complete rule-base, in which individual members of the population is mapped to a single rule, whereas the Pittsburgh approach represents a complete rule-base in a single chromosome. Both the Michigan [39] and Pittsburgh [37, 56] approaches have been applied to fuzzy systems.

The Michigan approach is limited mainly by a credit assignment problem, which focuses on assessing the performance of each individual. The credit assignment problem is largely the result of conflicts between cooperation in rule inference and competition in selection. The bucket-brigade algorithm is a conventional means for solving the credit assignment problem. Individuals in a population are members of a rule-base, making the classifier prone to forget the result of learning because certain critical rules (individuals) die. The Pittsburgh approach does not have a credit assignment problem because individuals compete rather than cooperate with each other. On the other hand, the Pittsburgh approach does have its limitations. The chromosome is always much larger than that in the Michigan approach because each individual includes a complete rule-base instead of single rule. Consequently, the large memory compromises evolutionary efficiency, making it difficult for the optimum solution to evolve. In general, the proposed polyploidy method can be treated as a unique Pittsburgh classifier since a polyploidy includes multiple rules. Moreover, a simple to complex evolutionary process is proposed to avoid the large memory problem of the Pittsburgh classifier.

The rest of this paper is organized as follows: section 2 introduces the details of the proposed fuzzy polyploidy method in modeling fuzzy systems and explains the *structural adaptation* mechanism. Section 3 demonstrates the feasibility of applying the proposed method to Mamdani and TSK types FLC, as well as discusses those results. Concluding remarks are made in section 4.

2. POLYPLOIDY

Diploidy and its more general form, polyploidy, are widespread throughout nature. The polyploidy biological design not only uses a group of haploids to encode biological traits, but also benefits survival. Polyploidy provides replicate encoding so that an individual has additional survival strategies, and the behavior or phenotype of an individual can alter after environmental changes. Polyploidy is accompanied with a *dominant* mechanism that was identified by Mendel in the 19*th* century. The dominant mechanism controls the phenotype of a polyploidy so that only one trait in a chromosome acts as the surrogate of the group at any given time. In addition to polyploidy, multiple chromosome structures are also found in nature. High level life forms generally contain a complex genetic structure with both multiple chromosomes and polyploidy. For example, humans have a multiple diploidy structure (23 diploidy). Fig. 1 illustrates the haploidy, polyploidy and multiple chromosomes. These biological methods provide further insight into the feasibility of the proposed model to simulate complex genetic structures found in nature.

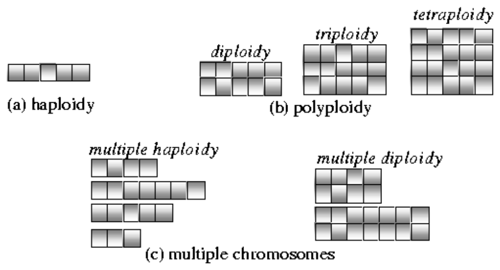


Fig. 1. Example of haploidy, polyploidy and multiple chromosomes.

For ease of implementation, conventional genetic algorithms use the simplest encoding form, haploidy, to represent the possible solutions of a problem. However, this simplification restricts the effectiveness of a GA in developing a more complex or robust solution. The solution of certain real world problems varies according to different problem solving states. For instance, in playing chess, a good player or computer program often adopts a strategy based on the circumstances of play. This example suggests that the polyploidy, rather than haploid, encoding should be adopted to represent possible solutions at certain problem solving states. Modeling a fuzzy system is comprised of structure identification and parameter identification. Structure identification initially selects an appropriate model for knowledge representation, such as a fuzzy rule-base or neural network, and then determines a suitable capacity, such as the number of rules, depending on the behavior of the application. Although some studies suggest analyzing the distribution of previous training data to roughly estimate a reasonable rule number or feature space partitioning [6], structure identification still relies largely on human decisions and is difficult to integrate with parameter identification. The second stage, parameter identification, optimizes the parameter values of a fuzzy system, such as the shape of the fuzzy membership functions and the contents of a fuzzy rule-base. In polyploidy encoding each rule in a fuzzy system is represented as a monoploidy and each polyploidy contains a complete rule-base. Thus, owing to the ability of a rule-base to completely transform into its corresponding polyploidy, identifying both the structure and parameter of a fuzzy system is equivalent to optimizing the performance of the polyploidy.

In nature, a dominant mechanism controls the phenotype of a polyploidy. Natural dominance determines the surrogate when certain units in the polyploidy include distinct traits. In contrast, the proposed approach treats relationships between polyploid chromosomes as relationships of cooperation rather than of dominance. Since the problem solving states are fuzzy rather than crisp, fuzzy membership can be used to represent the cooperative relationship. Restated, each chromosome is part of a weighted combination that is guided by fuzzy membership functions. To distinguish it from natural dominance, in this paper this mapping is referred to as *fuzzy dominance*.

Fuzzy dominance can be implemented in two ways. One, fixed fuzzy membership functions are used to partition the environment space equally as shown in Fig. 2. Thus, each polyploidy unit is considered to be responsible for certain problem solving states. On the other hand, the membership functions can be designed to have a particular shape based on the environment, as illustrated in Fig. 3. The most convenient means of adapting fuzzy membership is to incorporate the membership function into the chromosome encoding. Therefore, the membership function can be optimized via genetic evolution.

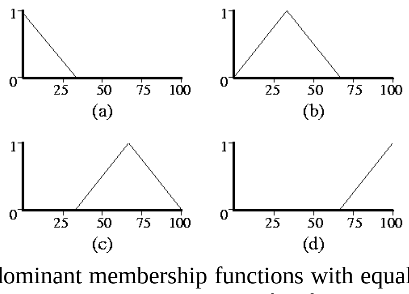


Fig. 2. Example of fuzzy dominant membership functions with equal partitioning for a tetraploidy. (a), (b), (c), (d) are dominant memberships for first, second, third and forth unit of a tetraploidy respectively.

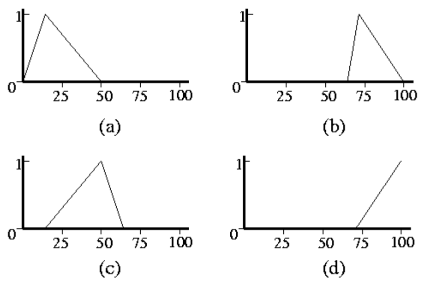


Fig. 3. Example of fuzzy dominance membership functions with arbitrary partitioning for a tetraploidy. (a), (b), (c), (d) are dominant memberships for first, second, third and forth unit of a tetraploidy respectively.

Membership functions can be represented and encoded in many ways [43, 56, 57]. Most of these approaches can be directly applied to the proposed model with only slight modification. We propose another example to encode triangular membership functions in polyploid.

$$Triangle(x;a,b,c)=\begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \tag{1}$$

As shown in Eq. 1, a triangular membership function *Triangle*(x; a, b, c) is specified by three parameters $\{a, b, c\}$, which denote the left, center and right corners of the triangle, respectively. Regarding the memory usage, only the center point of a triangular membership function is encoded, while the left and right corners are determined by the other center points. The left corner of a membership function is assigned to the largest among the center points which are less than current center, and the right corner is assigned to the smallest among the center points which are bigger than current center. To encompass the complete input space, the leftmost and rightmost membership functions are modified. The left corner of the leftmost membership function is set to $-\infty$, and the right corner of the rightmost membership function is set to $+\infty$.

Fig. 4 illustrates the membership function encoding of a polyploidy containing five units. In this case, the polyploid contains five genes for describing the center corners of the corresponding fuzzy membership functions. For easy explanation, we denotes these center corners as c_1, c_2, c_3, c_4, c_5, where $c_1 < c_2 < c_3 < c_4 < c_5$. The membership functions corresponding to c_1, c_2, c_3, c_4, c_5, are *Triangle* (x; $-\infty$, c_1, c_2), *Triangle* (x; c_1, c_2, c_3), *Triangle* (x; c_2, c_3, c_4), *Triangle* (x; c_3, c_4, c_5) and *Triangle* (x; c_4, c_5, ∞), respectively.

The simple to complex evolution process forms the basis for structural adaptation. According to natural evolution, life evolves from simple forms to more complex forms since life was originally only a unicellular protozoan or protophyta, such as an amoeba that consisted of protoplasm and organelles with very simple functions. Higher organisms consist of an enormous number of cells and have complex organs. Moreover, human learning also proceeds sequentially from simple to more complex knowledge.

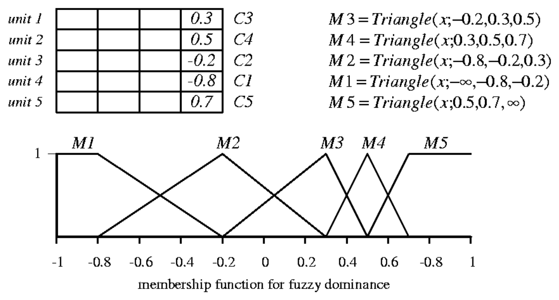

unit 1
unit 2
unit 3
unit 4
unit 5
0.3 C3
0.5 C4
-0.2 C2
-0.8 C1
0.7 C5
M3 = Triangle(x;−0.2,0.3,0.5)
M4 = Triangle(x;0.3,0.5,0.7)
M2 = Triangle(x;−0.8,−0.2,0.3)
M1 = Triangle(x;−∞,−0.8,−0.2)
M5 = Triangle(x;0.5,0.7,∞)
M1
M2
M3
M4
M5
1
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 -0.8 1
membership function for fuzzy dominance

Fig. 4. Adaptive fuzzy dominance. The Center apices of triangular membership functions are encoded in a polyploidy.

The simple to complex process is a vital aspect of the polyploidy model. Although providing a flexible encoding scheme and a more accurate solution than conventional GAs, polyploidy involves a large search space and large memory for encoding, thus compromising the search efficiency. Directly evolving polyploidy with a large number of units, say, ten units, is likely to fail to attain the optimum. To overcome this difficulty, we recommend evolving the polyploidy structure sequentially, just like simple to complex evolution in nature. The initial population is restricted to having the simplest structure, and the survivors has a certain probability of varying their structures. Structural variation helps the evolution explore new structures, thus allowing better structures to be found. Restated, the simple to complex mechanism first identifies the most vital strategy or a solution with a limited simple structure to survive, and then attempts to develop a more accurate solution with a complex structure.

As the foundation of evolution proceeds from simple to complex, structural variation is essential to maintaining the structural diversity of a population. Structural variation in the proposed model includes two opposite functions, structural expansion and structural simplification. Structural expansion complicates the structure, whereas structural simplification makes the structure simple and concise. Therefore, structural evolution moves toward two opposite directions with equal chance, thus preventing the structure from over expanding.

Fig. 5 illustrates the proposed structural expansion and structural simplification. The structural expansion inserts one unit into a polyploidy at unit boundary. As mentioned earlier, each unit represents a rule including antecedence and consequence, and a polyploidy represents a complete rule-base. Structural expansion is equivalent to inserting a new rule into a rule-base. The antecedent (condition) part of a newly inserted unit is set randomly. To ensure that the performance of a new expanding polyploidy does not differ markedly from that of its parent, the consequence of a newly inserted unit is set as the conclusion of the parent rule-base performing under the circumstances in which the new inserted rule is completely firing. On the other hand, to maintain the conciseness of a rule-base, structural simplification deletes the unit that appears to be the most redundant. The deleted unit is determined by calculating the similarity of the antecedents, and then the unit with the highest similarity to others is selected for deletion.

As mutation diversifies the contents or gene value of a chromosome, the structural variation diversifies the structure of individuals. Structural variation can also be viewed as either a unique mutation or a macro mutation. The probability of each individual undergoing structural variation at a generation is defined here as a *structural variation rate*. A low structural variation rate limits the evolution's ability to explore new polyploidy structure, and a large variation rate is prone to interrupting the evolution. We recommend setting the structural expansion rate between 0.01 and 0.1.

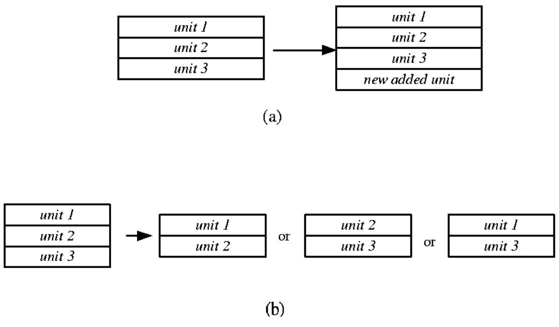


Fig. 5. Structural variation operations. (a) structural expansion, and (b) structural simplification.

In addition to the simple to complex mechanism, the design of genetic operators is another important issue in the proposed polyploidy method. Due to the possibility of a population containing individuals with distinct structures, the conventional mutation and crossover must be concerned with becoming a polyploidy method. The ability of the original mutation to simply change the context of certain genes is also found in polyploidy chromosome, thus making the polyploidy mutation operation fully compatible with previous methods. Next, consider the crossover. A conventional crossover cannot operate between polyploidy with distinct structures. To avoid this conflict, the crossover must be modified. Crossover between distinct structures can be accomplished by introducing a *coercion operation*, which simply converts two conflicting structures into identity. Thus, the polyploidy crossover first coerces the parents into identical structure, and then the parents can crossover as conventional crossover. According to Fig. 6, the coercion operation can be designed as a master-slave approach [18], in which one parent is selected as the master and the other, the slave. The coercion either expands or simplifies the slave structure that identifies with the master. Fig. 7 describes a situation in which the polyploidy crossover uses structural coercion.

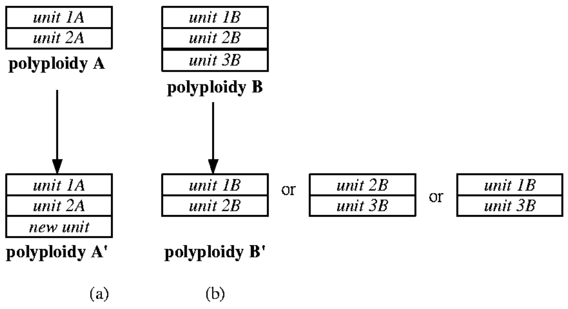


Fig. 6. Structural coercion. (a) selects polyploidy B as master, and (b) selects polyploidy A as master.

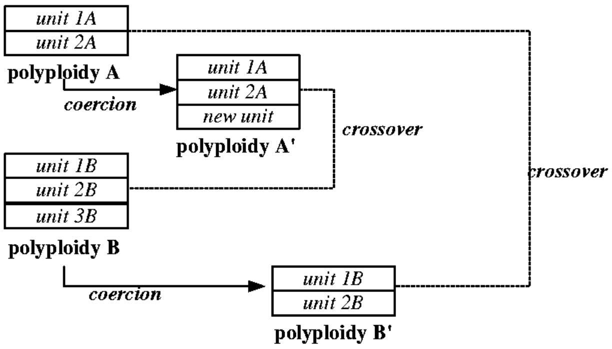


Fig. 7. Crossover between parents (polyploidy A and polyploidy B) with distinct structures.

3. EXAMPLE OF FUZZY MODELING

Modeling fuzzy systems that use polyploidy comprises three major stages, similar to the problem solving process of common GAs. First, the chromosome encoding is determined. Before a complete fuzzy system is encoded in a polyploidy, we must extract basic elements which compose the fuzzy system, and then represent the basic elements as monoploidies. Consider a fuzzy rule-bas: the monoploidy can be selected as the representation of a fuzzy rule. Consequently, a polyploidy consisting of a repeated monoploidy can represent a complete fuzzy rule-base since it consists of a set of fuzzy rules. In some applications with a hierarchical rule-base, such as a neural fuzzy system or fuzzy knowledge network, the encoding can be designed by introducing the multiple chromosome concept. Thus the node in the same hierarchy level is represented as a single polyploidy, and then those polyploids representing distinct hierarchy levels can compose a multiple polyploidy. However, the encoding design is closely related to an application. The encoding design largely rests on the principle that relational units in a polyploidy should be as close as possible; otherwise, the polyploidy would be prone to breaking up by a one-point crossover. The second and third stages for modeling a fuzzy system using polyploidy include designing a fitness measure and selecting adequate genetic operations. In these stages, polyploidy GAs do not differ from common GAs.

In this section, we present two examples of PI-like fuzzy logic controllers (FLC) using the polyploidy model. As is well known, scrambled rules are prone to a loss generality in a fuzzy system. To avoid scrambled rules, some studies directly encode all elements in a PI-like FLC matrix and imposed punishment for scrambled rules as judged by MacVicar-Whelan meta-rules in fitness evaluation [51]. We believe that doing so reduces the efficiency of evolution because most rules in the encoding space are scrambled and are not desired. As we know, too much learning ability provided by an adaptive FLC causes a scrambled rule-base, accounting for why a simplified PI-like FLC is adopted here to avoid this situation. Keeping concise rule-bases is important for an FLC, and concise rule-bases have the advantages of computational efficiency and generality. This work adopts the simplification based on MacVicar-Whelan rule-bases [58]. MacVicar-Whelan rule-bases provide the foundation for manually designing a comprehensive and meaningful rule-base for an FLC. The MacVicar-Whelan rule-bases can be simplified with little reasonable modification [59, 60]. Table 1 displays the simplified MacVicar-Whelan rule-base for an $N \times N$ FLC. This simplification reduces the number of rules for an N × NFLC from N^2 to $2 \times N$, and the simplified FLC still effectively solves low order control problems in many applications. The concise representation reduces the risk of scrambled rules. Although employing a fully matrix representation for an adaptive FLC creates a more accurate rule-base, doing so often scrambles the rule-base during learning. In addition to avoiding scrambled rules, using simplified rules has the advantages of a short chromosome, i.e., concise genetic encoding and small search space. Table 2, which is a generalized form of Table 1, presents the rule-base used in this research.

Table 1. The simplified MacVicar-Whelan fuzzy rule base.

RULE 1:	**IF** error is *Z*
	THEN change of control is *Z*
RULE 2:	**IF** change of control is *Z*
	THEN change of control is *Z*
RULE 3:	**IF** error is *NL*
	THEN change of control is *NL*
RULE 4:	**IF** change of error is *NL*
	THEN change of control is *NL*
RULE 5:	**IF** error is *NM*
	THEN change of control is *NM*
RULE 6:	**IF** change of error is *NM*
	THEN change of control is *NM*
RULE 7:	**IF** error is *NS*
	THEN change of control is *NS*
RULE 8:	**IF** change of error is *NS*
	THEN change of control is *NS*
RULE 9:	**IF** error is *PS*
	THEN change of control is *PS*
RULE 10:	**IF** change of error is *PS*
	THEN change of control is *PS*
RULE 11:	**IF** error is *PM*
	THEN change of control is *PM*
RULE 12:	**IF** change of error is *PM*
	THEN change of control is *PM*
RULE 13:	**IF** error is *PL*
	THEN change of control is *PL*
RULE 14:	**IF** change of error is *PL*
	THEN change of control is *PL*

Table 2. The fuzzy rule base we used.

RULE 1:	**IF** error is E_{11}

THEN change of control is C_{11}
RULE 2: **IF** change of control is E_{21}
THEN change of control is C_{21}
.
.
.
RULE 2n-1: **IF** error is E_{1n}
THEN change of control is C_{1n}
RULE 2n: **IF** change of control is E_{2n}
THEN change of control is C_{2n}

The original MacVicar-Whelan meta-rules are designed for Mamdani FLC [61]. However, we believe that the simplified FLC can also be applied to other FLCs, such as the TSK model [62, 63]. These FLCs differ in their representation of the consequence part of a rule. Based on the simplified PI-like FLC and the polyploidy, we pair the two simplified rules concentrating on different inputs in the FLC. Consequently, the paired rules become the basic element, i.e. unit, of the polyploidy chromosome. Because there are N pairs of rules for the simplified $N \times N$ FLC, various FLC structures can be represented as a fuzzy polyploidy. As mentioned earlier, a polyploidy consists of several monoploidy units with each unit sharing a homogeneous genetic encoding and phenotype mapping method. In this example, paired rules are represented as a monoploidy, and the simplified $N \times N$ FLC is represented as the polyploidy with n-units. The search space is reduced from N^2 to $2 \times N$.

In the following, we describe the experiment of the simplified Mamdani type FLC using the polyploidy model. An example is also provided showing how to implement the TSK model. In addition, two plant processes with second order transfer functions defined in [51] are tested. The transfer functions of the two processes, referred to as plant A and plant B (labeled plant B and plant C, respectively in [51]), are presented as follows:

$$PlantA : G_A(S) = \frac{2}{(S+1)\cdot(S+2)} \tag{2}$$

$$PlantB : G_B(S) = \frac{2}{S\cdot(1+0.1\cdot S)} \tag{3}$$

Figs. 8 and 9 display the closed-loop step responses of each plant. The test plants are transformed into a second order differential equation for ease of computer simulation. The differential equation is

$$PlantA:$$

$$2r(t) = c''(t) + 3c'(t) + 2c(t)$$

$$\Rightarrow \Delta \mathrm{c}_{\mathrm{k}} = \frac{T^2}{3T+1}\left(2r_{k-1} - 2c_{k-1} + \frac{\Delta c_{k-1}}{T^2}\right) \tag{4}$$

$PlantB:$

$$2r(t) = 0.1c''(t) + c'(t) \tag{5}$$

$$\Rightarrow \Delta c_k = \frac{T^2}{T+0.1}(r_{k-1} + \frac{0.1\Delta c_{k-1}}{T^2})$$

where T is a small constant time interval, c_k is the output signal at the kth time steps, r_k is the input signal at the kth time steps.

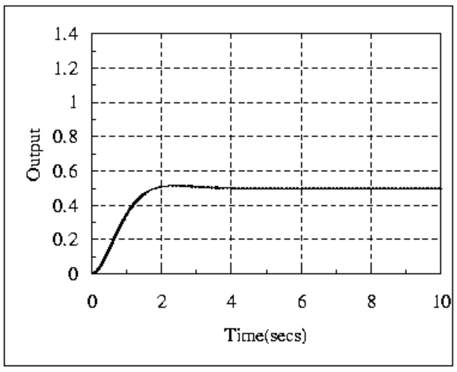


Fig. 8. Closed loop step responses for (plant A).

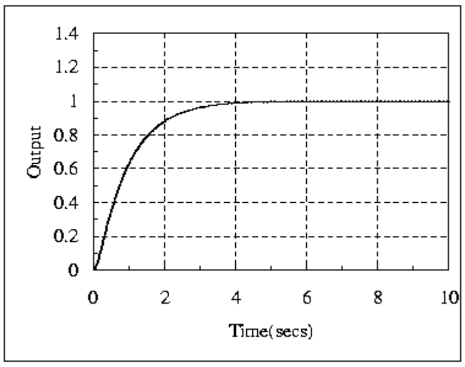


Fig. 9. Closed loop step responses (plant B).

The integral-of-time-multiplied absolute error (*ITAE*), shown in Eq. 6, is an appropriate measure for fitness. However, since GAs are typically used for maximization rather than minimization, the fitness $\frac{1}{ITAE}$ is defined as of the first ten seconds, as in Eq. 7.

$$ITAE: \int t|e(t)|dt \tag{6}$$

$$Fitness = \frac{1}{\int_0^{10} t \cdot |e(t)| dt} \tag{7}$$

A. Mamdani fuzzy models

Developed in 1975 and widely regarded as the most well-known fuzzy model, the Mamdani fuzzy model applies fuzzy theory to solve control problems by a set of linguistic control rules obtained from human experts. Fig. 10 illustrates the matrix representation of a typical Mamdani PI-like fuzzy model. Both antecedent and consequent parts of the control rules are expressed as fuzzy linguistic terms in a Mamdani model. Therefore each rule has a fuzzy output, and an extra defuzzification process is necessary to transform the output from fuzzy to crisp.

	PL	PM	PS	PZ	NZ	NS	NM	NL
PL	NL	NL	NL	NL	NL	NM	NS	NZ
PM	NL	NL	NM	NM	NM	NS	NZ	PS
PS	NL	NM	NS	NS	NS	NZ	PS	PM
PZ	NM	NM	NS	NZ	PZ	PS	PM	PM
NZ	NM	NM	NS	NZ	PZ	PS	PM	PM
NS	NM	NS	PZ	PS	PS	PS	PM	PL
NM	NS	PZ	PS	PM	PM	PM	PL	PL
NL	PZ	PS	PM	PL	PL	PL	PL	PL

Fig. 10. A Mamdani PI-like FLC. PL, PM, PS, PZ, NZ, NS, NM and NL are fuzzy memberships.

Among the several conventional approaches that use the defuzzification process, include centroid of area, mean of maximum and bisector of area. However, these approaches are too complex and time-consuming to use in a computer program. Using centroid defuzzification and sum-product composition [64] provides a more practical means of implementation. The crisp output in this approach is equivalent to the weighted average of centroids of consequent membership functions, and the weighted factor equals the firing strength of each rule multiplied by the area of the its consequent membership function [6], as given in Eq. 8, where w_i is the firing strength of the i^{th} rule, a_i and z_i are the area and the centroid, respetively, of the consequent membership of the i^{th} rule. This approach provides a more efficient means for achieving defuzzification. In a PI-like FLC, the crisp output in Eq. 8 is the change of the system outputs.

$$CrispOutput = \frac{w_1 \cdot a_1 \cdot z_1 + w_2 \cdot a_2 \cdot z_2 + \ldots + w_k \cdot a_k \cdot z_k}{w_1 \cdot a_1 + w_2 \cdot a_2 + \ldots + w_k \cdot a_k} \tag{8}$$

Fig. 11 illustrates a chromosome encoding of the Mamdani type PI controller mapping to Table 2. Each rule pair composing two rules is defined as a unit of a polyploidy, and the whole polyploidy corresponds to a complete Mamdani type rule-base. The two rules together relate the output error and change in output error. Encoding the antecedent for each rule is the center point of the triangular membership function as described in previous section. For ease of implementation, the consequence of each rule is restricted to a triangular membership function with the same height and bandwidth. Thus the membership functions have the same area. To reduce the cost of defuzzification, the centroid is directly encoded in the polyploidy instead of the center point of membership function. Therefore, defuzzification can be calculated only by the firing strength and centroid of the consequent membership function for each rule. Since the consequent of each rule has the same area, the crisp output can be calculated as Eq. 9.

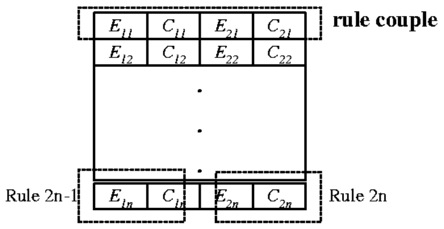

rule couple
E_{11} C_{11} E_{21} C_{21}
E_{12} C_{12} E_{22} C_{22}
Rule 2n-1
E_{1n} C_{1n} E_{2n} C_{2n}
Rule 2n

Fig. 11. Polyploidy encoding for a Mamdani FLC. E_{1i} is the center point of the membership function that relates the error of the output, while E_{2i} relates $\Delta error$, and C_{2i} is the centroid of the consequence membership function.

$$CrispOutput = \frac{w_1 \cdot z_1 + w_2 \cdot z_2 + \ldots + w_k \cdot z_k}{w_1 + w_2 + \ldots + w_k} \tag{9}$$

This work initially tested the plant A that uses fuzzy polyploidy encoding with a fixed polyploidy structure containing 20 units. Five simulations were run with different random seeds. The population size was set to 300, and the crossover rate was set to 0.5 or 50%. The probability of each gene undergoing mutation is 5%. Fig. 12 shows the evolution curves of each run. The average fitness values of each run at the 5000th and the 10000th generation are 9.48 and 9.66, respectively, and the evolution was extremely slow in exploring better population after 5000 generations. These experiments yielded acceptable results. Fig. 13 illustrates the time-response graphs of the best FLC found after 10000 generations of five runs with different random seeds. These results resemble those in Cheong and Lai's work [51]. However, the comparison is meaningless since the FLC size in Cheong and Lai's work differs from that in our work. Although the time-response graphs reveal that the best FLC of all runs can effectively control plant process A, the results of such 20 × 20 FLCs can be further improved, reducing converge time and response ripple.

In addition to a fixed structure, the previous test is extended by using a fuzzy polyploidy with structural adaptation of simple to complex evolution. Each member of the initial population has a structure with three units, and the structure upper bound during

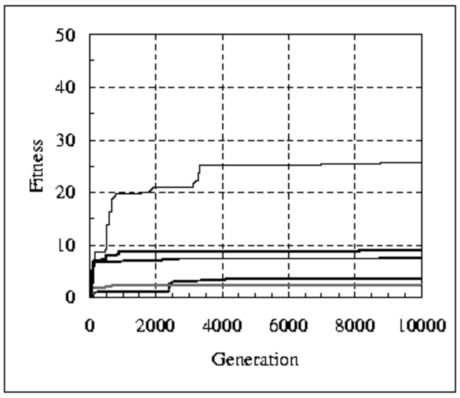


Fig. 12. Evolution curves of polyploidy GA with fixed structure in optimizing plant A.

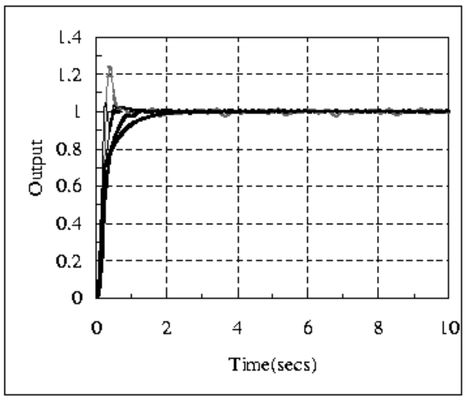


Fig. 13. Best FLCs for plant A obtained by polyploidy GA with fixed structure.

evolution was set to 20 units. The structural variation rate was set to 3%, and the probabilities of evolution toward structural expansion and structural simplification were set equal. Thus, both the probabilities of structural expansion and structural simplification were 1.5%. Five simulations were also run with different random seeds; these experiments are referred to as simulations *A* to *E*. Fig. 14 displays the evolution curves of each run. The average fitness values of each run at the 5000*th* and the 10000*th* generation are 12.50 and 15.31, respectively, which are higher than in the previous case without structural adaptation. However, simulation E did not produce satisfactory results.

Comparing simulation E with the other simulations reveals that simulation E was trapped on a structure with two units and seven units, and did not further find better individuals via structural expansion. The structural migration graphs of simulations *A* and *E*, as displayed in Fig. 15 and Fig. 16, confirm this condition. The structural migration graph displays the average structural distribution of each generation. Where y-axis denotes the evolution time (generation), and the x-axis represents the various polyploidy

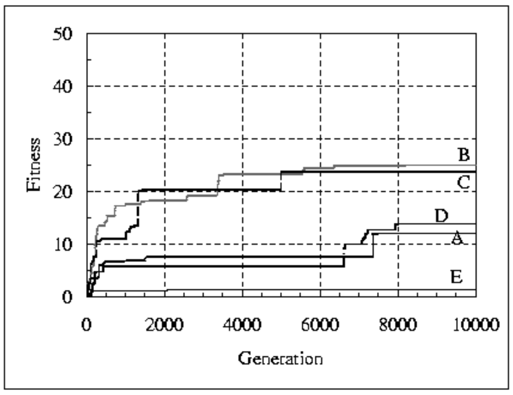


Fig. 14. Evolution curves of polyploidy GA with structural adaptation in optimizing plant A.

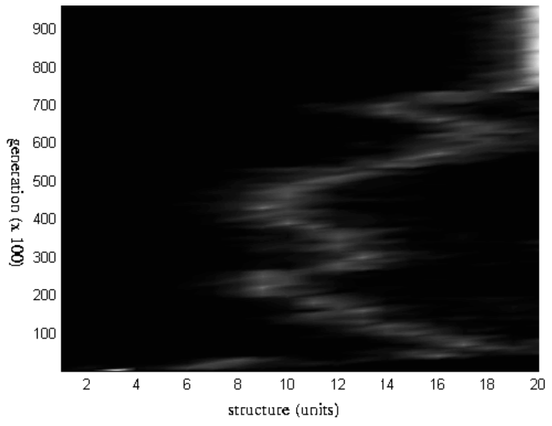


Fig. 15. Structural migration graph of simulation A in learning plant A

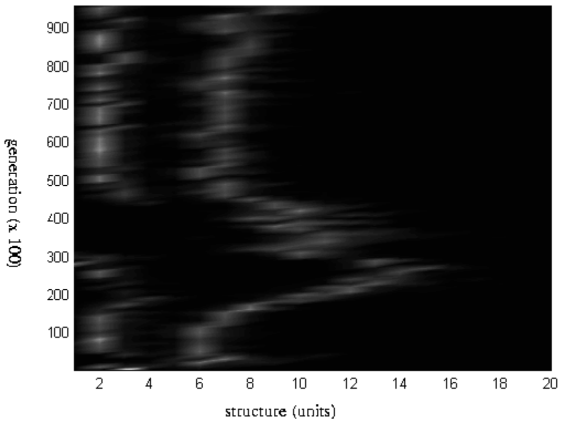


Fig. 16. Structural migration graph simulation E in learning plant A.

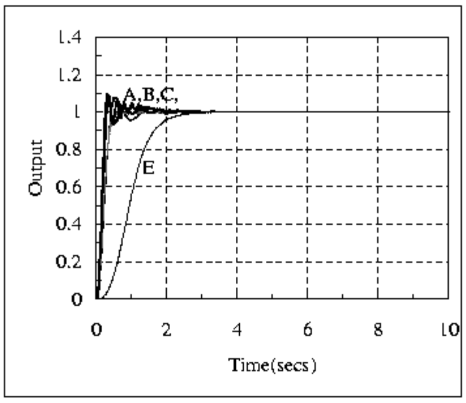


Fig. 17. Best FLCs for plant A obtained by polyploidy GA with structural adaptation.

structures from one to ten units. Different colors on the graph refer to the individual count of a given structure at a particular generation, with the lighter color representing a higher individual density. According to Fig. 16, simulation E fails to explore a new structure with a better performance other than the two and ten units. This condition can be improved by increasing the structural variation rate. Fig. 17 shows the time-response graph of the best FLC of each run at the 10000*th* generation. According to this figure, the response rapidly converges to the target gain 1 in simulations *A* to *D*, and is much better than the previous case without structural adaptation.

The experiments of plant B obtained satisfactory results. Two experiments use polyploidy with and without structural adaptation to yield even better results than those of Cheong and Lai. Figs. 18 and 19 display the time-response graph of the best FLC found at 10000*th* generation using polyploidy with and without structural adaptation. Both experiments smoothly and rapidly converge as seen in the figures. However, the structural adaptation did not obviously outperform those cases without it. Fig. 20 displays the structural migration graph of simulation E with structural adaptation. Notably, the simulation is worse than most cases largely because the structural changes are perhaps too fast, leading to inadequate generation for individuals to explore a sufficient number of genes at shorter structures. The results above demonstrate that the proposed polyploidy model also has a structural premature convergence problem. To avoid structural premature convergence, the structural variation should be carefully designed, and controlling the structural variation rate by techniques such as self-adaptation and simulated annealing would possibly be helpful.

B. TSK fuzzy models (Sugeno fuzzy models)

In addition to the Mamdani fuzzy models, TSK fuzzy models are also used extensively because their ease of defuzzification. Table 3 illustrates a typical TSK model with two inputs and a single output. The consequence part of each rule in a TSK model is formulated as a crisp function of its input. The system output, given in Eq. 10, is the

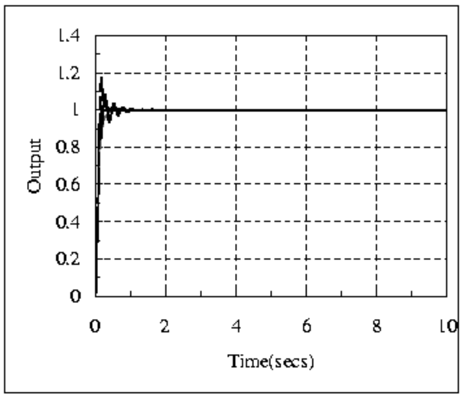


Fig. 18. Best FLCs for plant B obtained by polyploidy GA with fixed structure.

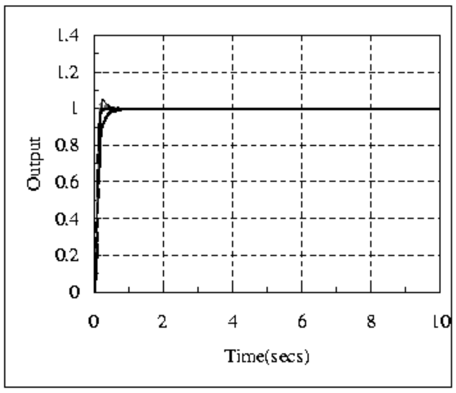


Fig. 19. Best FLCs for plant B obtained by polyploidy GA with structural adaptation.

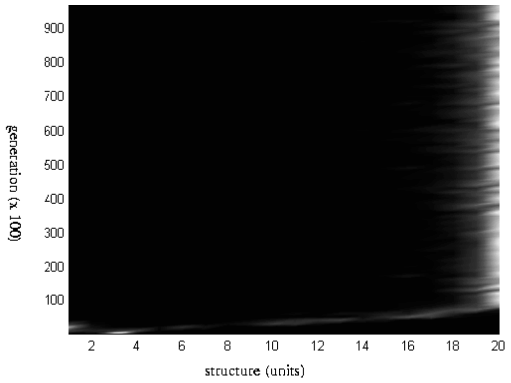


Fig. 20. Structural migration graph simulation E in learning plant B.

weighted average of the consequence of each rule based on its firing strength, where w_k is the firing strength of each rule and z_k is the output value of each consequence function. Therefore, TSK models directly generate a crisp output without an extra defuzzification process.

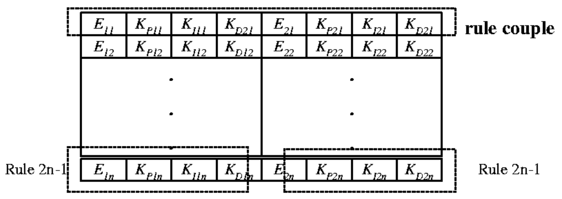
Table 3. A TSK model.

RULE 1:	**IF** X is *small* **AND** Y is *small*
	THEN $z = X - Y + 5$
RULE 2:	**IF** X is *small* **AND** Y is *large*
	THEN $z = X + Y - 2$
RULE 3:	**IF** X is *large* **AND** Y is *small*
	THEN $z = -X - Y + 3$
RULE 4:	**IF** X is *large* **AND** Y is *large*
	THEN $z = -X - Y - 6$

$$CrispOutput = \frac{w_1 \cdot z_1 + w_2 \cdot z_2 + \ldots + w_k \cdot z_k}{w_1 + w_2 + \ldots + w_k} \tag{10}$$

The chromosome encoding of TSK models resembles that of Mamdani models except for the consequent part. Fig. 21 shows the polyploidy encoding of a PI-like first order TSK fuzzy logic controller. The consequent part of each rule in TSK models is encoded as the weights relative to the output error and change in output error. Each rule produces a crisp output by calculating the weighted sum as in Eq. 11. Moreover, the output of each rule is equivalent to a local PID controller, and the system output concludes the crisp output of each local controller according to the firing strength as given by Eq. 12.

We omit the experiments since the previous simulation can also be treated as a zero-order TSK model, which has only constant consequence. A higher order TSK model can generate more complex behavior than a zero-order one, and should obtain similar or better results than the previous experiment.



E_{11} K_{P11} K_{I11} K_{D21} E_{21} K_{P21} K_{I21} K_{D21}
rule couple
E_{12} K_{P12} K_{I12} K_{D12} E_{22} K_{P22} K_{I22} K_{D22}
Rule 2n-1
E_{1n} K_{P1n} K_{I1n} K_{D1n} E_{2n} K_{P2n} K_{I2n} K_{D2n}
Rule 2n-1

Fig. 21. Polyploidy encoding for a TSK FLC. E_{1i} is the center point of the membership function that relates the error of the output, and E_{2i} relates Δ error. K_P, K_I and K_D are the parameters of the output polynomial for each rule.

$$z_i = K_P \cdot \Delta error + K_I \cdot error + K_D \tag{11}$$

$$\Delta output = \frac{w_1 \cdot z_1 + w_2 \cdot z_2 + \ldots + w_k \cdot z_k}{w_1 + w_2 + \ldots + w_k} \qquad (12)$$

4. CONCLUSIONS

This work provides a novel approach to modeling fuzzy systems. The proposed polyploidy encoding provides a more flexible and dynamic encoding scheme than previous methods. Conventionally, genetic algorithms have difficulty obtaining optimum solutions when the chromosome structure is too complex, because a complex structure always involves a large search space. The proposed simple to complex process improves on both effectiveness and efficiency for evolving inside a large search space involving a complex structure. This study also develops two new genetic operations, structural simplification and structural expansion. These operations provide the structural variation ability of the polyploidy model. The conventional crossover operation is modified to become the condition of a population comprised of various structures. The proposed approach not only makes it easier to achieve evolution in a complex structure, but also provides a convenient means of automatically identifying the structure and parameters of a fuzzy system in a single step. Moreover, examples of Mamdani and TSK FLC designs, which present a more concise encoding of fuzzy rules, reduce the search space of an $N \times N$ FLC from N^2 to $2 \times N$. Since it is concise, this encoding also avoids scrambled rules owing to its conciseness.

Simulations indicate that, although the proposed model is effective, the structural variation rate must be carefully controlled. We believe that the variation rates, mutation rates, performance and search space are all related. Future work should more closely examine these relations. To further improve the polyploidy model, techniques such as self adaptation or simulating annealing can be used to adapt the structural variation rate. On the other hand, the polyploidy model can be further combined with other techniques such as symbolic evolution and co-evolution as proposed in [52] to increase the efficiency of evolution. Further studies are needed to more fully explore these issues.

REFERENCES

1. L. A. Zadeh, “Fuzzy sets,” *Information and Control*, Vol. 8, 1965, pp. 338-353.
2. J. A. Freeman, *Neural Networks*, Addison Wesley, 1991.
3. R. J. Schalkoff, *Artificial Neural Networks*, McGraw-Hill, 1997
4. D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, 2000.
5. L. A. Zadeh, “Soft computing and fuzzy logic,” *IEEE Software*, 1994, Vol. 11, pp. 48-56.
6. J.-S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*, Prentice Hall, 1997.
7. D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence: A Logical Approach*, Oxford University Press, 1998.
8. C. Cercone and G. McCalla, “Ten years of computational intelligence,” *Computa-*

tional Intelligence, Vol. 10, 1994, pp. i-vi.
9. S. Mitra and Y. Hayashi, "Neuro-fuzzy rule generation: survey in soft computing framework," *IEEE Transactions on Neural Networks*, Vol. 11, 2000, pp. 748-768.
10. J. J. Buckley and Y. Hayashi, "Fuzzy neural networks: A survey," *Fuzzy Sets and Systems*, Vol. 66, 1994, pp. 1-13.
11. M. M. Gupta and D. H. Rao, "On the principles of fuzzy neural networks," *Fuzzy Sets and Systems*, Vol. 61, 1994, pp. 1-18.
12. D. Nauck, F. Klawonn, and R. Kruse, *Foundations of Neuro-Fuzzy Systems*, Wiley, 1997.
13. Y. Hayashi, J. J. Buckley, and E. Czogala, "Fuzzy neural network with fuzzy signals and weights," in *Proceedings of International Joint Conference on Neural Networks*, 1992, pp. 696-701.
14. J.-S. R. Jang, "Anfis: Adaptive network based fuzzy inference systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, 1993, pp. 665-685.
15. C. T. Lin and C. P. Jou, "GA-based fuzzy reinforcement learning for control of a magnetic bearing system," *IEEE Transactions on Systems, Man, Cybernetics*, Vol. 30, 2000, pp. 276-289.
16. L. J. Fogel, "Toward inductive inference automata," in *Proceedings of the International Federation for Information Processing Congress*, 1962, pp. 395-400.
17. J. H. Holland, "Adaptive plans optimal for payoff-only environments," in *Proceedings of the Second Hawaii International Conference on System Sciences*, 1969, pp. 917-920.
18. J. H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.
19. I. Rechenberg, *Evolutionsstrategie: Optimierung Technischer System nach Prinzipien der Biologischen Evolution*, Frommann-Holzboog, 1973.
20. H.-P Schwefel, "Evolutionsstrategie und numerische optimierung," PhD Thesis, Dept. of Process Engineering, Technical University of Berlin, Germany, 1975.
21. J. R. Koza, "Genetically breeding populations of computer programs to solve problems in artificial intelligence," in *Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence*, 1990, pp. 6-9.
22. J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
23. J. R. Koza, *Genetic Programming: Automatic Discovery of Reusable Programs*, MIT Press, 1994.
24. J. H. Holland, "Genetic algorithms and classifier systems: foundations and future directions," *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, 1987, pp. 82-89.
25. T. Bäck, U. Hammel, and H. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, Vol. 1, 1997, pp. 3-17.
26. T. Bäck and H. Schwefel, "Evolutionary computation: an overview," in *Proceedings of IEEE Conference on Evolutionary Computation*, 1996, pp. 20-29.
27. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley, 1989.
28. M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1996.

29. T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford Univ. Press, 1996.
30. O. Cordón and F. Herrera, "A general study on genetic fuzzy systems," *Genetic Algorithms in Engineering and Computer Science*, Wiley, 1995, J. Periaux, G. Winter, M, Galan, and P. Cuesta ed., pp. 33-57.
31. O. Cordón, F. Herrera, and M. Lozano, "A classified review on the combination fuzzy logic-genetic algorithms bibliography: 1989-1995," *Genetic Algorithms and Fuzzy Logic Systems, Soft Computing Perspectives*, World Scientific, 1997, E. Sanchez, T. Shibata, and L. Zadeh, ed., pp. 209-241.
32. S. Matsushita, T. Furuhashi, H. Tsutsui, and Y. Uchikawa, "Efficient search for fuzzy models using genetic algorithm," *Information Sciences*, Vol. 110, 1998, pp. 41-50.
33. C. Karr, "Genetic algorithms for fuzzy controllers," *AI Expert*, Vol. 6, 1991, pp. 26-33.
34. C. L. Karr and E. J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Transaction on Fuzzy Systems*, Vol. 1, 1993, pp. 46-53.
35. D. Park, A. Kandel, and G. Langholz, "Genetic-based new fuzzy reasoning models with application to fuzzy control," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, 1994, pp. 39-47.
36. F. Bolata and A. Nowé, "From fuzzy linguistic specifications to fuzzy controllers using evolution strategies," in *Proceedings of 4th International Conference on Fuzzy Systems*, 1995, pp. 1089-1094.
37. F. Herrera, M. Lozano, and J. L. Verdegay, "Tuning fuzzy logic controllers by genetic algorithms," *International Journal of Approximate Reasoning*, Vol. 12, 1995, pp. 299-315.
38. P. Thrift, "Fuzzy logic synthesis with genetic algorithms," in *Proceedings of 4th International Conference on Genetic Algorithms*, 1991, pp. 509-513.
39. A. González and R. Pérez, "Structural learning of fuzzy rules from noised examples," in *Proceedings of 1995 IEEE International Conference on Fuzzy Systems*, 1995, pp. 1323-1330.
40. M. Valenzuela-Rendon, "The fuzzy classifier system: a classifier system to continuously varying variable," in *Proceedings of Fourth International Conference on Genetic Algorithms*, 1991, pp. 346-353.
41. M. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proceedings of Second IEEE International Conference on Fuzzy System*, 1993, pp. 612-617.
42. A. Homaifar and E. Maccormick, "Simultaneous design of membership function and rule sets for fuzzy controllers using genetic algorithms," *IEEE Transactions on Fuzzy System*, Vol. 3, 1995, pp. 129-139.
43. J. Liska and S. S. Melsheimer, "Complete design of fuzzy logic systems using genetic algorithms," in *Proceedings of International Conference on Fuzzy Systems*, 1994, pp. 1377-1382.
44. A. Parodi and P. Bonelli, "A new approach of fuzzy classifier systems," in *Proceedings of Fifth International Conference on Genetic Algorithms,*1993, pp. 223-230.
45. K. Kiguchi, K. Watanabe, K. Izumi, and T. Fukuda, "Application of multiple fuzzy-neuro force controllers in an unknown environment using genetic algorithms," in *Proceedings of the 2000 IEEE International Conference on Robotics and Automation*, 2000, pp. 2106-2111.

46. R. J. Streifel, R. J. Marks II, R. Reed, J. J. Choi, and M. Healy, "Dynamic fuzzy control of genetic algorithm parameter coding," *IEEE Transactions on Systems, Man, and Cybernetic-Part B: Cybernetics*, Vol. 29, 1999, pp. 426-433.
47. W. Pedrycz, "Genetic algorithms for learning in fuzzy relational structures," *Fuzzy Sets and Systems*, Vol. 69, 1995, pp. 37-52.
48. F. Herrera, M. Lozano, and J. L. Verdegay, "A learning process for fuzzy control rules using genetic algorithms," *Fuzzy Sets and Systems*, Vol. 100, 1998, pp. 143-158.
49. O. Cordón and F. Herrera, "A two-stage evolutionary process for designing TSK fuzzy rule-based systems," *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, Vol. 29, 1999, pp. 703-715.
50. Y. H. Joo, H. S. Hwang, K. B. Kim, and K. B. Woo, "Fuzzy system modeling by fuzzy partition and GA hybrid scheme," *Fuzzy Sets and Systems*, Vol. 86, 1997, pp. 279-288.
51. A. F. Gómez-Skarmeta and F. Jiménez, "Fuzzy modeling with hybrid systems," *Fuzzy Sets and Systems*, Vol. 104, 1999, pp. 199-208.
52. F. Cheong and R. Lai, "Constraining the optimization of a fuzzy logic controller," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, 2000, pp. 31-46.
53. B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approac*h, Prentice Hall, 1991.
54. C. F. Juang, J. Y. Lin, and C. T. Lin, "Genetic reinforcement learning through symbiotic evolution for fuzzy controller design," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 30, 2000, pp. 290-302.
55. D. E. Moriarty and R. Mikkilainen, "Efficient reinforcement learning through symbiotic evolution," *Machine Learning*, Vol. 22, 1996, pp. 11-32.
56. J. H. Holland and J. S. Reitman, "Cognitive systems based on adaptive algorithms," *Pattern-directed Inference Systems*, Academic Press, 1978.
57. S. F. Smith, "A learning system based on genetic adaptive algorithms," PhD Thesis, Dept. of Computer Science, University of Pittsburgh, 1980.
58. B. Carse, T. C. Fogarty, and A. Munro, "Evolving fuzzy rule base controllers using genetic algorithms," *Fuzzy Sets and Systems*, Vol. 80, 1996, pp. 273-293.
59. I. Dumitrache and C. Buiu, "Genetic learning of fuzzy controllers," *Mathematics and Computer in Simulation*, Vol. 49, 1999, pp. 13-26.
60. R. R. Yager and D. P. Filev, *Essentials of Fuzzy Modeling and Control*, Wiley, 1994.
61. P. J. MacVicar-Whelan, "Fuzzy sets for man-machine interactions," *International Journal of Man-Machine Studies*, Vol. 8, 1976, pp. 687-697.
62. M. Mizumoto, "Fuzzy controls under various approximate reasoning methods," in *Proceedings of 2nd IFSA Congress*, 1987, pp. 143-146.
63. E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, Vol. 7, 1975, pp. 1-13.
64. T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Transactions on System, Man and Cybernetics*, Vol. 15, 1985, pp. 116-132.
65. M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, Vol. 28, 1988, pp. 15-33.

Ming Da Wu (吳明達) was born in Taipei, Taiwan, in 1969. He received the B.S. degree in computer science from the Taung-Hai University, Taiwan, in 1993, and received the M.S. degree in computer science from the National Chiao-Tung University, Taiwan, in 1995. He is currently pursuing the Ph.D. degree at National Chiao-Tung University, Taiwan. His research interests include evolutionary computation, fuzzy systems, neural networks, and game theorem.

Chuen-Tsai Sun (孫春在) received his B.S. degree in electrical engineering (1979) and his M.A. degree in history (1984), both from National Taiwan University, Taiwan. He received his Ph.D. degree in computer science from the University of California at Berkeley in 1992. From 1991 to 1992 he was with the Lawrence Livermore National Laboratory, California, where he participated in research projects on fuzzy neural networks. Since 1992 he has been with National Chiao Tung University, Taiwan, where currently he is a Professor and the chairman of Department of Computer and Information Science. He is engaged in research and teaching in the areas of neuro-fuzzy systems, evolutionary computing, computer-assisted learning, and web-based distance education. He is also in charge of two community universities sponsored by the Hsinchu city government.