*Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (2003), *17*, 335–347. Printed in the USA.
Copyright © 2003 Cambridge University Press 0890-0604/03 $16.00
DOI: 10.1017/S0890060403174057

# Iterative genetic algorithm for learning efficient fuzzy rule set

MENG HIOT LIM AND WILLIE NG
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
(RECEIVED February 28, 2003; ACCEPTED January 4, 2004)

**Abstract**

We present a methodology of learning fuzzy rules using an iterative genetic algorithm (GA). The approach incorporates a scheme of partitioning the entire solution space into individual subspaces. It then employs a mechanism to progressively relax or tighten the constraint. The relaxation or tightening of constraint guides the GA to the subspace for further iteration. The system referred to as the iterative GA learning module is useful for learning an efficient fuzzy control algorithm based on a predefined linguistic terms set. The overall approach was applied to learn a fuzzy algorithm for a water bath temperature control. The simulation results demonstrate the effectiveness of the approach in automating an industrial process.

**Keywords:** Fuzzy Rules; Genetic Learning; Iterative Genetic Algorithm; Plant Automation; Process Control

## 1. INTRODUCTION

For industrial processes, fuzzy rule based systems (FRBSs) as a whole can be viewed as an approach that attempts to capture an operator or process engineer's domain specific expertise. These systems have become increasingly popular in the practical applications of artificial intelligence. One reason for such popularity can be attributed to the fact that fuzzy systems do not require analytic modeling of the task or problem being addressed. Unlike classical logic that requires a rigid classification of objects and decisions into one category or another, fuzzy logic offers a more flexible approach by admitting imprecision and vagueness in classification. Such flexibility makes it convenient to incorporate a human's expert knowledge. It is common for such knowledge to be represented as a set of linguistic rules by means of *IF*⟨*antecedents*⟩*THEN*⟨*consequents*⟩ conditional statements. The rules serve as an abstraction of the experience or knowledge about the system.

One major drawback of FRBSs is that they are not able to learn, but require that the knowledge be derived by a human expert. The development of a fuzzy control system typically requires the declaration of the rules based on heuristics derived through observations and experience pertaining to the problem domain. Therefore, it is common to manually declare the if–then rules and the membership functions. When developing a fuzzy system, the experts need to examine all input–output associations for the problem and then encode their experiences as linguistic rules. However, as system complexity increases, reliable fuzzy rules and membership functions for describing the system's behavior become difficult to determine. It is therefore beneficial to seek ways to bypass the need for the manual declarative approach.

The hybridization of fuzzy and genetic computing methodologies has become popular in recent years (Cordon et al., 2001). Many researchers, such as Hoffmann and Pfister (1997) and Magdalena (1998), have developed ways to design and implement fuzzy controllers using genetic algorithms (GAs). GAs have been used mainly for designing membership functions, for generating fuzzy if–then rules, and for simultaneous rule generation and membership function tuning. One of the earliest works on the hybridization of GAs and fuzzy systems can be found in Karr (1991), who used a GA to generate membership functions for the cart–pole balancing problem. However, this approach requires manual declaration of an exhaustive rule set. An exhaustive rule set consists of rules that logically associate a suitable course of action with all possible states of the fuzzy input variables.

Reprint requests to: M.H. Lim, School of Electrical and Electronic Engineering (EEE), Block S1, Nanyang Technological University, Singapore 639798. E-mail: emhlim@ntu.edu.sg



https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

By contrast, if we have predefined fuzzy membership conditions over the domain of a fuzzy variable and a fixed rule structure (decision table), a GA can be used to search for the best rule base, a strategy presented by Thrift (1991). He described the design of a two-input one-output fuzzy controller for centering a cart on a frictionless one-dimensional track. Homaifar and McCormick (1995) extended the approach to determine the rules and the base lengths of the triangular membership functions. Their work demonstrated that a GA could be employed for the simultaneous design of membership functions and rule sets for fuzzy logic controllers.

In our approach (Lim et al., 1996; Ng & Lim, 2002), the focus of research is on a genetic fuzzy system for automated generation of rules. It is assumed that a set of linguistic values is given for each input and output variable. This means that a fuzzy partition of the input and output space is given. We allow the GA to derive an *n*-rule fuzzy system based on predefined membership functions for the fuzzy terms. The if–then rules are coded as a string that represents the system's rule set. The task is to derive *n* input–output associations, whereby the input corresponds to the antecedent while the output corresponds to the consequent part of the fuzzy if–then rule. The value *n* becomes a constraint for the GA in the process of learning. By searching only *n* fuzzy rules, a significant reduction of the GA solution space is expected and thus the computational efficiency of GA is improved. In general, an *n*-rule fuzzy system may not necessarily be efficient because it may incorporate rules that are redundant. It is therefore possible to reduce the number of rules. We propose an iterative genetic algorithm learning module (IGALM) to optimize the *n*-rule fuzzy system in terms of *n* in our current work. The overall system of learning is demonstrated for an industrial process, the water bath temperature control system.

This paper is organized as follows. A brief summary of some of the relevant concepts in GA and FRBS as well as the genetic coding is presented in Section 2. Section 3 describes the GA learning paradigm. In order to demonstrate the applicability of our methodology, Section 4 presents a computer simulation of the IGALM approach on the temperature control system. Finally, Section 5 gives the conclusions on the IGALM method presented.

## 2. GENETIC–FUZZY LEARNING SYSTEM

In this section, the coding of a fuzzy system as a structure for GA learning will be described. The architecture and structure of a fuzzy system is first discussed. This is followed by a brief overview on GA before describing the coding structure.

### 2.1. Fuzzy system overview

A fuzzy system can be defined as a nonlinear system whereby one specifies the desired input/output behavior by defining the memberships and rules. It can imitate the control actions of human operators through description of the system behavior using linguistic expressions (e.g., high, low, medium). The algorithm that is implemented in this system consists of a set of if–then rules with linguistic values and variables. The general expression of the fuzzy rule can be written as

$$\text{IF} \quad x_1 = A_{1j}, x_2 = A_{2j} \cdots \quad \text{AND} \quad x_g = A_{gj},$$

$$\text{THEN} \quad y = D_j, \tag{1}$$

where $j$ indexes a set of linguistic values, $x = (x_1, x_2, \ldots, x_g)$ is a $g$-dimensional input vector, $A_{ij}$ is the antecedent linguistic value for input variable $x_i$, $y$ is the output variable, and $D_j$ is the conclusion linguistic value of the $j$th rule. Usually, the development of a fuzzy system involves defining a finite set of labels to represent the linguistic values for describing each variable. In general, if we let the number of labels be $\omega_1, \omega_2, \ldots, \omega_g$ to describe the input variables $x_1, x_2, \ldots, x_g$, respectively, the maximum number of rules one can declare will be $\omega_1 \times \omega_2 \times \cdots \times \omega_g$. This is referred to as the maximum or exhaustive rule set. As for an *n*-rule fuzzy system, it will have *n* number of rules, with *n* being less than or equal to $\omega_1 \times \omega_2 \times \cdots \times \omega_g$. This we refer to as an *n*-rule constrained fuzzy system or simply an *n*-rule fuzzy system.

### 2.2. GA overview

GAs are search algorithms based on natural selection, a biological adaptive behavior that espouses the mechanism of "survival of the fittest." They carry out stochastic evolutionary processes to achieve global optimization through the use of selection, crossover, and mutation operators. The search space of the problem at hand is represented as a collection of individuals, which are often referred to as chromosomes. Each chromosome is a potential solution for the problem. A GA associates each potential candidate solution in the population with a fitness measure to evaluate the quality of the solution. Selection is achieved by favoring fitter chromosomes so as to ensure that useful chromosomal properties are carried forward to the next generation. As in nature, survival and propagation into future generations depend on mechanisms of guided selection. In this manner, weaker chromosomes are removed. This way, the search exhibits a form of exploitative behavior. Chromosomes that are fit in the current population are exploited in order to guide the search toward the optimum.

In addition to exploitation, a GA incorporates mechanisms to explore other regions in the search space. Crossover and mutation are the recombination operators for this purpose. Crossover combines two chromosomes (parents) to produce a new chromosome (offspring). A new chromosome tends to be fitter than both the parents because it inherits desirable characteristics from both of them. Conversely, mutation randomly alters one or more gene values

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

| $\frac{dError}{dt}$ \ Error | NL | ZE | PL | PM | PH |
|---|---|---|---|---|---|
| NL | LI | LI | HI | | |
| ZE | HD | NC | LI | HI | HI |
| PL | | LD | NC | LI | HI |
| PM | | | LD | NC | LI |
| PH | | | | LD | NC |

**Fig. 1.** A 17-rule fuzzy system for controlling the temperature of a water bath plant.

in a chromosome. By altering the genes, the GA is able to maintain population diversity, a feature necessary for effective search. The mutation rate is usually fairly low. With a high mutation rate, the algorithm approaches that of an unguided random search. It is generally regarded that the success of a GA depends very much on the ability to strike an adequate balance between exploration and exploitation. Such a balance is usually achieved through a set of GA parameters such as population size, mutation rate, and crossover probability, which are tuned to achieve effective search performance.

### 2.3. Coding

We let $m$ represents the maximum number of genes in the string as well as the size of the maximum rule set. Consider $S_{n/m}$ to be the genetic structure of the string describing an $n$-rule fuzzy system, where $n$ represents the number of rules that the algorithm is required to uncover. Hence, $S_{n/m}$ is the string or chromosome, which can be written as

$$S_{n/m} = (g_1, g_2, g_3, \ldots, g_m). \tag{2}$$

The symbol $g_i$ is known as a gene. Each gene has its own position in the string and the value of a gene is an identifier corresponding to the consequent of the rule. This chromosome can be obtained easily from a fuzzy decision table or linguistic matrix (Thrift, 1991; Lim et al., 1996). Figure 1 depicts a $5 \times 5$ rule base represented as a decision table. It defines a 17-rule fuzzy system for the water bath temperature control system. The 17-rule fuzzy system is designed using a manual declarative approach. It has two input conditions, "error" and "rate of error change," and one output response. An unfilled cell indicates the absence of a term for the output variable. Here, we use a "NULL" label to define such a situation. For an $n$-rule fuzzy system, the number of NULL entries is equal to $(m - n)$. A chromosome can thus be obtained from the decision table by going through each cell in a row-wise manner and coding each output fuzzy set as an integer or as symbols. By applying the genetic coding to the table, one obtains the string "4450013455023450023400023". The integers 1, 2, 3, 4, 5, and 0 correspond to the output linguistic terms HD, LD, NC, LI, HI, and NULL, respectively.

The $S_{n/m}$ is a positional coding structure, which means that the functionality of a gene is determined by its position. The number of genes is equal to the total number of cells in the table. A specific location in the chromosome can thus be traced to a particular entry in the decision table.

It is also equally descriptive to represent the $S_{n/m}$ string as a structure as shown in Figure 2, where $\mathcal{L}$ denotes the locus of the gene in the conventional coded string and **A** corresponds to the allele of that gene or the consequent of the rule. Within the structure, the position arrangement for each gene is arbitrary but should not be repeated. For example, at position 2 of the old string, $\mathcal{L}_{\mathbf{A}}$ is equal to $2_5$ and it can be placed at any position along the new string. The NULL entries can be ignored. Thus, the length of the string is $n$ instead of the total number of cells. Such a representation facilitates discussion on the crossover mechanism.

## 3. LEARNING PARADIGM

In order to search for an efficient rule set for the system, one approach would be to incorporate it as a search objective in the fitness function. This is achieved by penalizing chromosomes with large number of rules and by rewarding chromosomes with lesser rules. Incorporating the rule size as a search objective explicitly adds another dimension of complexity to the solution space. This is illustrated in Figure 3, which shows the distribution of a $5 \times 5$ linguistic matrix solution space. Each point in the graph represents a solution subspace starting from a single rule to 25-rule system. A solution subspace contains the total number of chromosomes having the same number of fuzzy rules. Therefore,

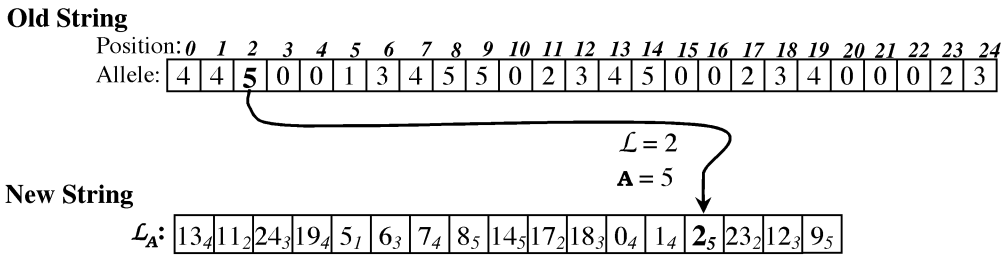



**Fig. 2.** The $\mathcal{L}_{\mathbf{A}}$ coding of fuzzy rules (see text for details).

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

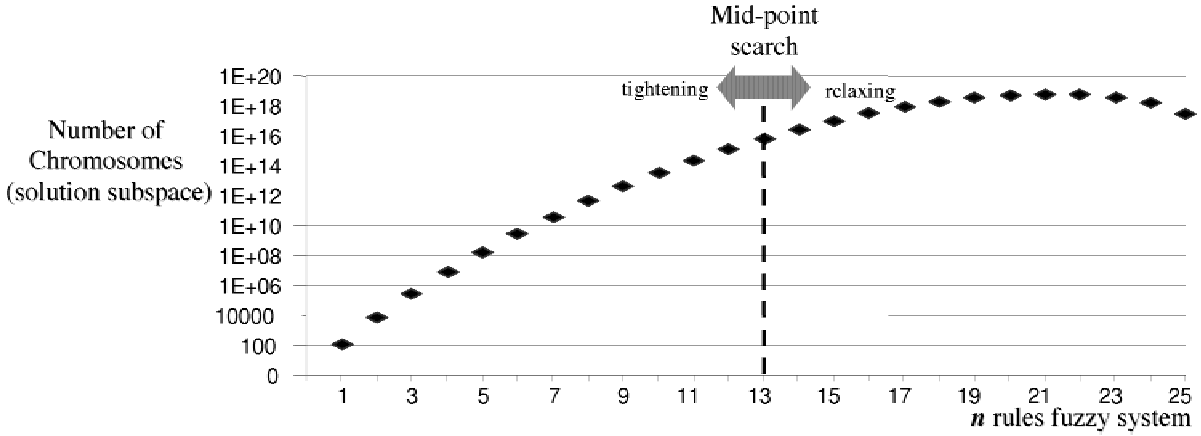



**Fig. 3.** The state space distribution for a 5 × 5 decision table.

the entire search space can be obtained by summing all the solution subspaces as follows:

$$\text{total search space} = \sum_{n=1}^{m} \alpha_n, \tag{3}$$

where $m$ denotes the maximum rule set, $n$ represents the number of rules of each chromosome and $\alpha_n$ represents a subspace. Each subspace corresponds to a search for a system with $n$ number of rules. The solution subspace can be written as

$$\alpha_n = {}^{m}C_n \times v^n, \tag{4}$$

where $v$ is the number of labels used to describe the output variable and ${}^{m}C_n$ refers to the notation for binomial coefficient. Consider a two-input system in which five linguistic values are used to describe the output with NULL entry in the matrix to denote the absence of a rule. If we sum up all the subspaces by substituting Eq. (4) into (3), the total search space will then be equal to $2.84 \times 10^{19}$. Thus, the search process might become prohibitively inefficient and slow if we employ the GA to handle the entire search space.

It should be noted that the search for the efficient rule set requires a mechanism to evaluate each potential solution. The chromosome evaluation is an integral component of GA learning. For some problems, the process of evaluation can be computationally "expensive." It is therefore evident that a global search of the entire solution space has a tendency to be computationally intractable. The next section addresses this issue by outlining the iterative GA approach.

### 3.1. Iterative GA learning

To overcome the potential computational complexity, the iterative GA incorporates a mechanism for systematically sieving through each solution subspace. From earlier discussion, it was clearly illustrated that the entire search space can be conveniently partitioned into individual subspaces based on the size (number of rules) of fuzzy rule set. With this understanding, the main procedures of the IGALM can be succinctly stated as follows:

1. Divide the entire solution space into subspaces.
2. Choose a subspace to start the iteration.
3. Employ a GA to derive a solution within the targeted subspace.
4. If the solution is found, repeat step 3 by tightening the constraint.
5. If the solution is not found, relax the constraint and repeat step 3.
6. Terminate the search once an acceptable solution is uncovered.

In the context of IGALM, the constraint refers to $n$, the size of the rule set. There are various options that one can adopt in deciding on the subspace to start the iteration. We list below four viable options in starting the GA iteration:

- Start with an optimistic performance target, where $n$ is equal to the minimum rule set.
- Start with pessimistic performance target, where $n$ is equal to the maximum rule set.
- Start from the middle, where $n$ is equal to half of the maximum rule set.
- Start at random, where $n$ is equal to any number between the minimum and maximum rule set.

The decision table mentioned in the previous section has been widely used in the design of a fuzzy system. Each entry in the table defines a fuzzy rule or input–output mapping. Depending on the nature of the application, it may not be necessary to evaluate every possible input combination because some may rarely or never occur. As for the case of the manually designed water bath control system, 17 rules out of the maximum 25 rules are being used (Fig. 1). There-

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

fore, in our approach, we assume a start in the middle option. We begin our search with $n$ being equal to half the size of the maximum rule set.

The functional flow chart of the IGALM is presented in Figure 4. It starts with a population of randomly created chromosomes. Each member of this population represents a potential solution for the problem, which is evaluated on the basis of the fitness evaluation function. Such a function is usually user defined and problem specific. The GA operators such as selection and crossover in the *generation loop* help to ensure that desirable properties of the chromosome are propagated into future generations. In general, the likelihood of fitter chromosomes being selected as parents to produce offspring by crossover is higher. Mutation is then applied to the offspring. The reproduction process is repeated until the desired number of offspring has been obtained to form the new generation of solutions. For each new generation, the chromosomes are again evaluated to determine their fitness level.

In this algorithm, $n$ can either be *tightened* or *relaxed*. We shall first discuss the former occurrence. Whenever the fitness value of a chromosome reaches a prespecified fitness threshold, the iteration of the generation loop breaks. This implies that a workable solution is found. It then proceeds to conquer new search area with $n$ being reduced by one. The GA will then restart to search for a new set of rules based on the latest value of $n$. In this way, the size of the fuzzy rule set shrinks as $n$ reduces. This process is repeated

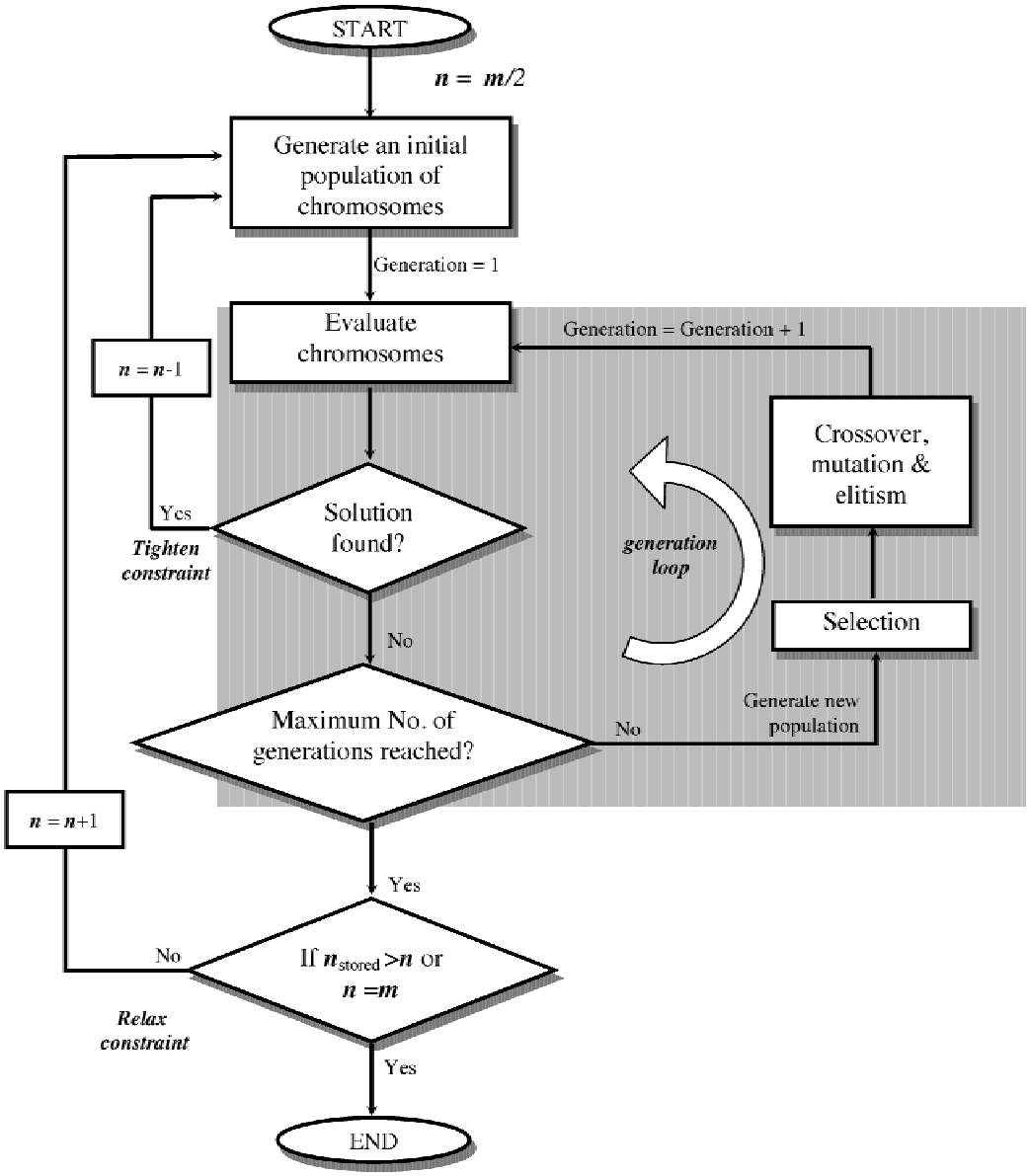



**Fig. 4.** The functional block architecture of IGALM.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

until the smallest possible value of $n$ is found. Hence, if optimum convergence does not occur and the maximum generation is reached, the search ceases. The end result will be the fuzzy rule set obtained based on the previous $n$ constraint. On the contrary, the relaxation of $n$ occurs if the GA fails to find a solution satisfying the initial $n$ constraint. The GA needs to explore new search space that is based on higher $n$ value. The increment will stop once a solution is found or $n$ has reached the maximum rule set $m$.

### 3.2. Selection and fitness scaling

A variety of selection schemes can be employed in GAs. For our implementation, we rely on the most common proportionate selection scheme, roulette wheel selection. This scheme is similar to spinning a roulette wheel with each slot sized proportionately to the fitness of each chromosome in the population. Thus, the individuals with higher fitness have better chances of being selected relative to the weaker individuals.

Often it is necessary to carry out fitness scaling for effective selection and reproduction. The purpose is to avoid premature convergence. Premature convergence arises during the early stages of a GA search process, whereby the existence of "super" chromosomes in a population overwhelms individuals of mediocre fitness. This usually results in convergence toward a local optimum. The fitness scaling applied is based on the linear scaling model in Goldberg (1989). This linear scaling technique significantly reduces the tendency for premature convergence as chromosomes of very high fitness are scaled down while the lowly members of the population get scaled up. The scaled fitness values of the chromosomes will serve as a basis for selecting parent chromosomes for reproduction. Details of this scaling technique can also be found in Lim et al. (2000).

### 3.3. Crossover

As mentioned earlier, each gene in the chromosome represents a rule of the fuzzy system. With a $n$-rule constraint imposed on the IGALM, $n$ number of rules are randomly distributed at various positions in the string. It is important to ensure that any crossover operation used does not violate the constraint so that the resultant offspring will contain exactly $n$ rules. Therefore, simple one-point or two-point crossover operators are not suitable. To illustrate this, consider two strings $S1$ = "0100510345" and $S2$ = "5340130010", representing two configurations of a six-rule system of maximum rule set 10.

By applying a two point-crossover at positions 1 and 3, we obtain a seven-rule system for $S1'$ and a five-rule system for $S2'$ (see Fig. 5). This clearly shows that traditional crossover operators are unable to satisfy the general $n$-rule constraint.

In order to overcome this problem, we employ a technique referred to as position-assigned crossover (PAX). PAX is similar to the partially matched crossover operator described by Goldberg and Lingle (1985). To demonstrate the operation of PAX, consider again the strings, $S1$ and $S2$. The basic steps for the crossover of $S1$ and $S2$ are as follows:

1. Apply the $\mathcal{L}_{\mathbf{A}}$ representation scheme.
2. Arbitrarily select $S1$ or $S2$ to be the reference string.
3. Rearrange $S2$ so that the genes with common $\mathcal{L}$ values are aligned.
4. Randomly choose two points and perform a two-point crossover.
5. Ensure that the offspring $S1'$ and $S2'$ satisfy the $n$-rule constraint.

The PAX operation is illustrated in Figure 6. Here, $S1$ and $S2$ undergo a two-point PAX at crossover positions 1 and 4. Clearly, there is no violation of the $n$-rule constraint because the offspring that are produced have the same number of rules as their parents.

### 3.4. Mutation and Elitism

Mutation is an operator that changes the allele of a gene randomly. It helps to prevent the GA from stagnating at local optima as well as to enhance exploration of the search space. In the population, the probability of a chromosome being selected depends on the mutation rate $P_m$. Mutation usually occurs at a very low rate. When selected, one of the $n$ genes in the selected chromosome will be chosen randomly and mutated.

An illustration of the mutation operation is given in Figure 7, where $S1'$ is an offspring taken from previous section. If this string is selected, one of its $\mathcal{L}_{\mathbf{A}}$ genes will undergo mutation. Two types of mutation can be considered. The $\mathcal{L}_{\mathbf{A}}$ gene can be altered either by using an $\mathcal{L}$-type mutation or

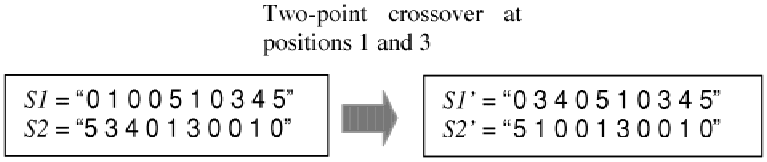



**Fig. 5.** A two-point crossover.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

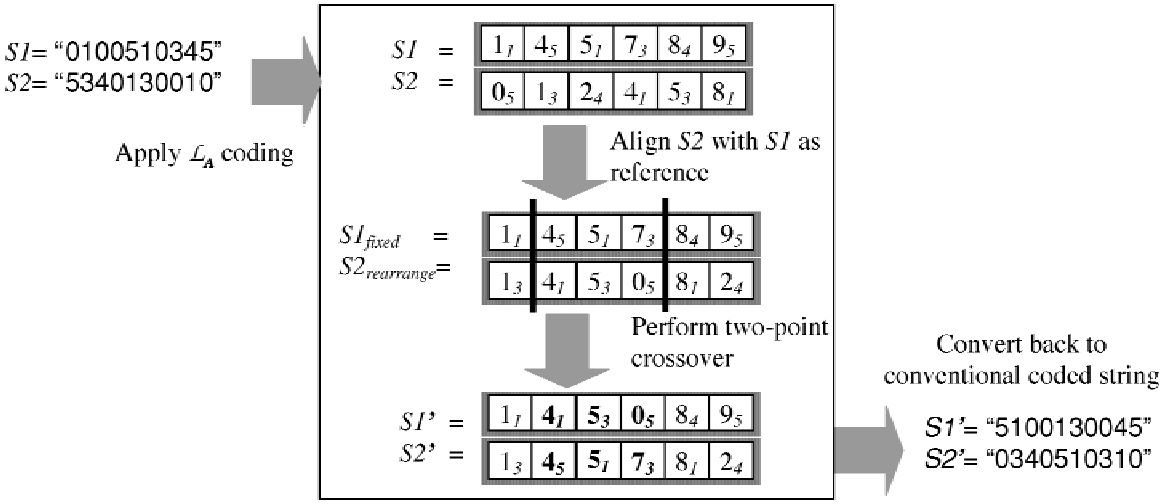



**Fig. 6.** A two-point position-aligned crossover.

an **A**-type mutation. In the $\mathcal{L}$-type mutation, occasional perturbations are introduced into the pool of chromosomes. As for the **A**-type mutation, the value of **A** is either incremented or decremented (Thrift, 1991). Similar to the crossover operator, we need to ensure that mutation does not result in violation of the *n*-rule constraint. In general, $\mathcal{L}$-type mutation results in the exploration of new area and **A**-type mutation creates small disturbances in the regional area. Because the impact of an $\mathcal{L}$-type mutation on a chromosome's properties is significantly higher than an **A**-type mutation, we set its frequency of occurrence to be 3 times lower than that of an **A**-type mutation.

Another feature of the IGALM is the use of elitism. When creating a new population by crossover and mutation, it is possible that we lose the best chromosome. Elitism is realized through preservation of the best member in the population. It is a method whereby a selected number of the best chromosomes in the population are copied without changes to the new population. Elitism can very rapidly increase the performance of a GA, because it prevents the loss of the best found solution.

## 4. IMPLEMENTATION

In this section, we describe the GA learning paradigm as applied to the temperature control of an industrial process. We aim to let the IGALM automatically derive a set of rules to control the temperature of the water bath.

### 4.1. Setup

Several algorithms have already been described in the literature for the temperature control system (Khalid & Omatu, 1992; Lin et al., 1999). In our work, we make use of the plant model presented in Tanomaru and Omatu (1992). The simulated control plant model in discrete time form is described as follows:

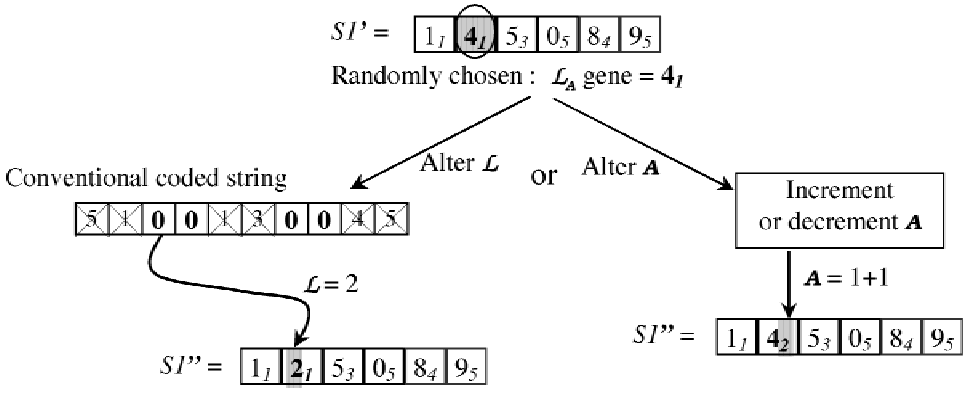



**Fig. 7.** The $\mathcal{L}$- and **A**-type mutations.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

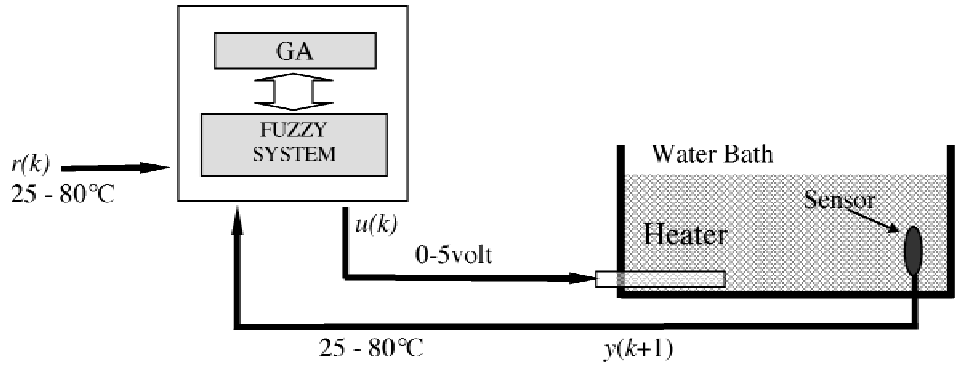



**Fig. 8.** The overall setup of the water bath temperature control system.

$$y(k+1) = a(T_S)y(k) + \frac{b(T_S)}{1+e^{0.5y(k)-\gamma}}\,u(k) + [1-a(T_S)]Y_0, \quad (5)$$

where $a(T_s) = e^{-\alpha Ts}$ and $b(T_s) = (\beta/\alpha)(1 - e^{-\alpha Ts})$. The water plant parameters in this simulation are $\gamma = 40.0$, $\alpha = 1.0151\text{E}{-4}$, $\beta = 8.67973\text{E}{-3}$, and room temperature $Y_0 = 25°\text{C}$. The schematic diagram of the overall setup is shown in Figure 8, where $r(k)$ is the target temperature. The objective is to control the temperature of the water bath within the range of 25 to 80°C. The heater is powered by the output signal $u(k)$ from the fuzzy system. The water temperature increases as the voltage increases. The value of $u(k)$, which depends on the desired temperature $r(k)$ and the plant's current temperature $y(k + 1)$, is described by fuzzy control rules. Functionally, the role of the GA is to derive a fuzzy system with $n$ rules.

### 4.2. Fitness function

The fitness function is a key feature in the selection process. It evaluates a chromosome to determine its fitness relative to the other chromosomes in the population. The GA uses this information to evolve better chromosomes over many generations. Therefore, the success of the algorithm in getting a good solution depends significantly on the design of a good fitness function. The choice of a good fitness function is specific to the problem at hand. Intuition often helps in formulating the fitness function.

When formulating the fitness function, the search objectives of the problem domain must first be identified. In order to define these objectives, a typical temperature response of the water bath system is examined. In Figure 9, we assume $T$ represents the total operating time of the plant, $T_{\text{rise}}$ denotes the number of time steps for the system to reach the desired temperature $r(k)$, and $T_{\text{stable}}$ denotes the

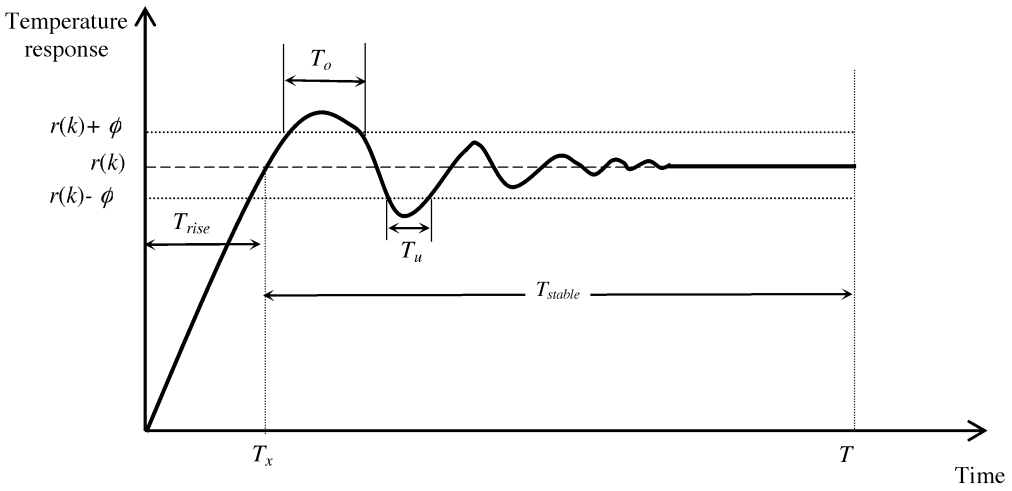



**Fig. 9.** The typical temperature response of a water bath system.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

number of time steps from the time when the system reaches $r(k)$ until $T$. The symbols $T_o$ and $T_u$ represent the overall number of time steps elapsed when the system overshoots, or undershoots respectively. The symbol $\phi$ is used to represent the temperature tolerance, which is the acceptable fluctuation or deviation from $r(k)$.

For the water bath control, we identified two objectives for the fitness function. The first is that a good control is where the rule set is able to react to the temperature change within a short time. This implies that $T_{rise}$ must be as low as possible. However, too fast a response does not guarantee a stable system. An acceptable $T_{rise}$ is one that will also allow the temperature to settle to the desired set point in the shortest possible time. The second objective we defined pertains to overshoot or undershoot after reaching $r(k)$. For this purpose, we further define $T_{out}$ to be the sum of $T_o$ and $T_u$. A low $T_{out}$ is generally desirable, as it implies greater temperature stability. With these two objectives, we aim to achieve a deadbeat control response. A deadbeat response is one that proceeds rapidly to the desired level and holds at that level with minimal overshoot. The two objectives, $T_{rise}$ and $T_{out}$, can be defined as follows:

$$T_{rise} = \sum_{j=1}^{T_x} \Theta_j, \quad \Theta_j = \begin{cases} 0 & y(k) > r(k), \\ 1 & \text{otherwise}, \end{cases} \tag{6}$$

$$T_{out} = \sum_{j=T_x}^{T} \Omega_j, \quad \Omega_j = \begin{cases} 0 & [r(k) - \phi] \le y(k) \le [r(k) + \phi], \\ 1 & \text{otherwise}. \end{cases} \tag{7}$$

The overall block architecture of the chromosome evaluator is as shown in Figure 10. Because each chromosome represents an entire fuzzy rule set, the fitness function must measure the overall effectiveness of the entire rule set. As shown in the diagram, the inference engine uses the chromosome to regulate the temperature of the water bath. In order to assess the performance of the fuzzy rule set, it is tested with different initial conditions. The initial conditions are based on various combinations of $r(k)$ and $y(k)$ within the range of control. For each test, the temperature

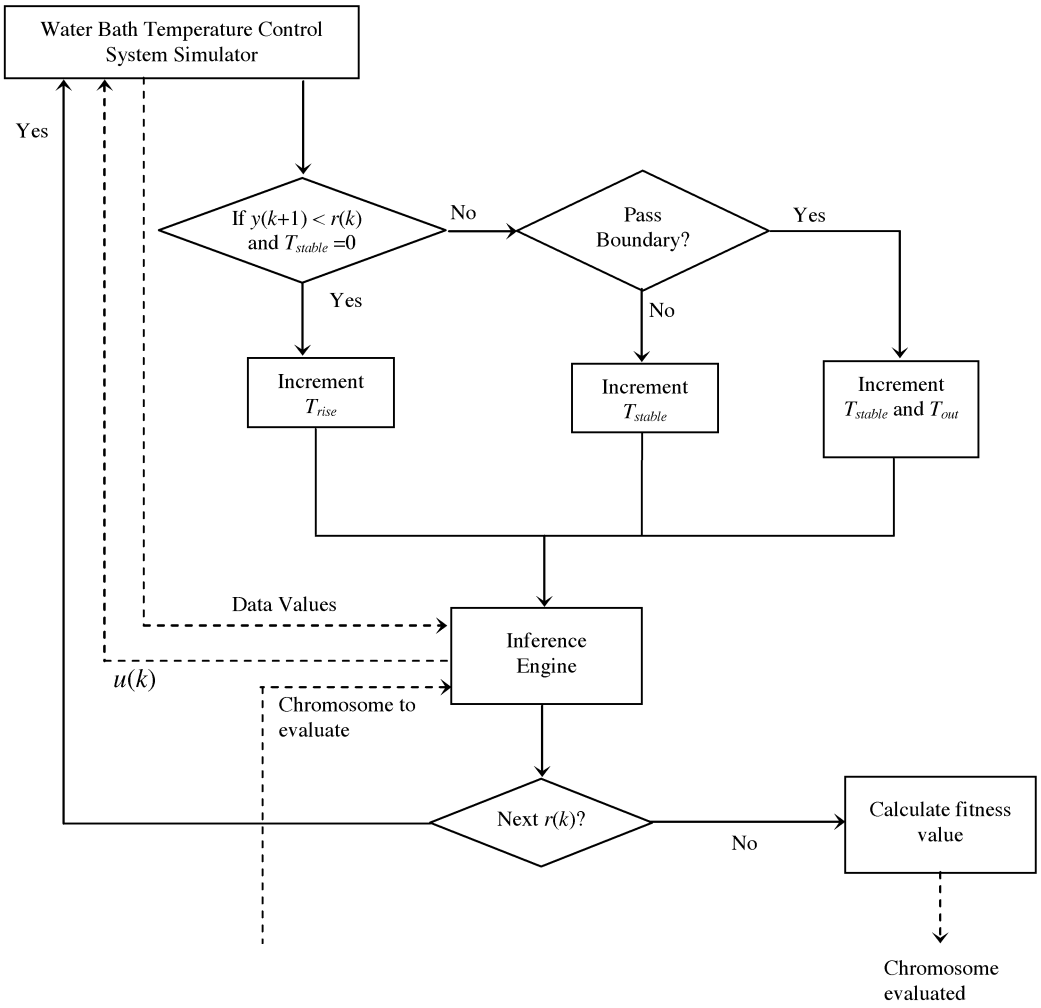



**Fig. 10.** The block architecture of the chromosome evaluator.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

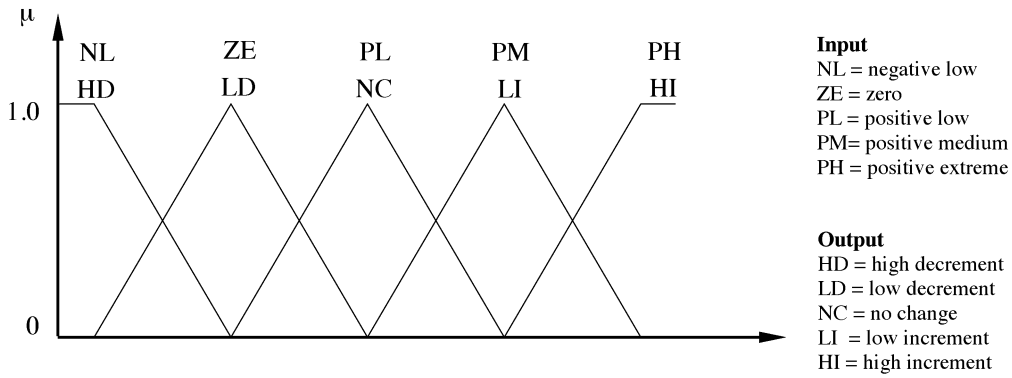



**Fig. 11.** The membership function for Error, $d\text{Error}/dt$ and $u(k)$.

control system is simulated until the prespecified number of $T$ time steps is reached. The fitness function based on the two objectives described is as follows:

$$f = \sum_{i=1}^{d} \left( \frac{a(T_{\text{stable}} - T_{\text{rise}})}{T_{\text{out}} + 1} \right). \tag{8}$$

The fitness is derived by subjecting the chromosome to $d$ number of control evaluations. Each evaluation corresponds to a randomly chosen target temperature. To favor chromosomes with fast response, a weighting parameter $a$ is included in the equation. Because the total $T$ is a sum of $T_{\text{rise}}$ and $T_{\text{stable}}$, a shorter $T_{\text{rise}}$ implies a longer $T_{\text{stable}}$, hence a higher fitness for the chromosome. However, if $T_{\text{rise}}$ is greater then $T_{\text{stable}}$, $f$ is set to zero. As mentioned earlier, in order to achieve our search objectives, $T_{\text{out}}$ must be kept as low as possible. Thus, $T_{\text{out}}$ is included in the fitness function as a penalty for chromosomes producing high overshoot or undershoot. With the above fitness function, a high fitness value implies small $T_{\text{rise}}$, large $T_{\text{stable}}$, and minimal overshoot or undershoot.

### 4.3. Simulation results

Three settings of targeted temperature were tested on the water bath temperature control system: 25–40, 40–60, and 60–80°C. The computation of rise time, settling time, and extent of overshoot or undershoot was carried out for each of the temperature ranges. For each range, the simulation was conducted over 60 samples with a sampling time $T_s$ of 30 s. Each test run therefore corresponds to an equivalent real time of 90-min duration.

For the IGALM, the system parameters are a population size $(m)$ of 100, a mutation rate $(P_m)$ of 0.03, a crossover probability $(P_c)$ of 0.6, and a fitness scaling factor $(S_f)$ of 2. The fitness function weighting constant $(a)$ is set to 3. The membership functions for the input variables Error and $d\text{Error}/dt$ and output variable $u(k)$ are shown in Figure 11. The two inputs, Error and $d\text{Error}/dt$, are defined as follows:

$$\text{Error} = r(k) - y(k+1), \tag{9}$$

$$\frac{d\text{Error}}{dt} = \frac{y(k+1) - y(k)}{T_s}. \tag{10}$$

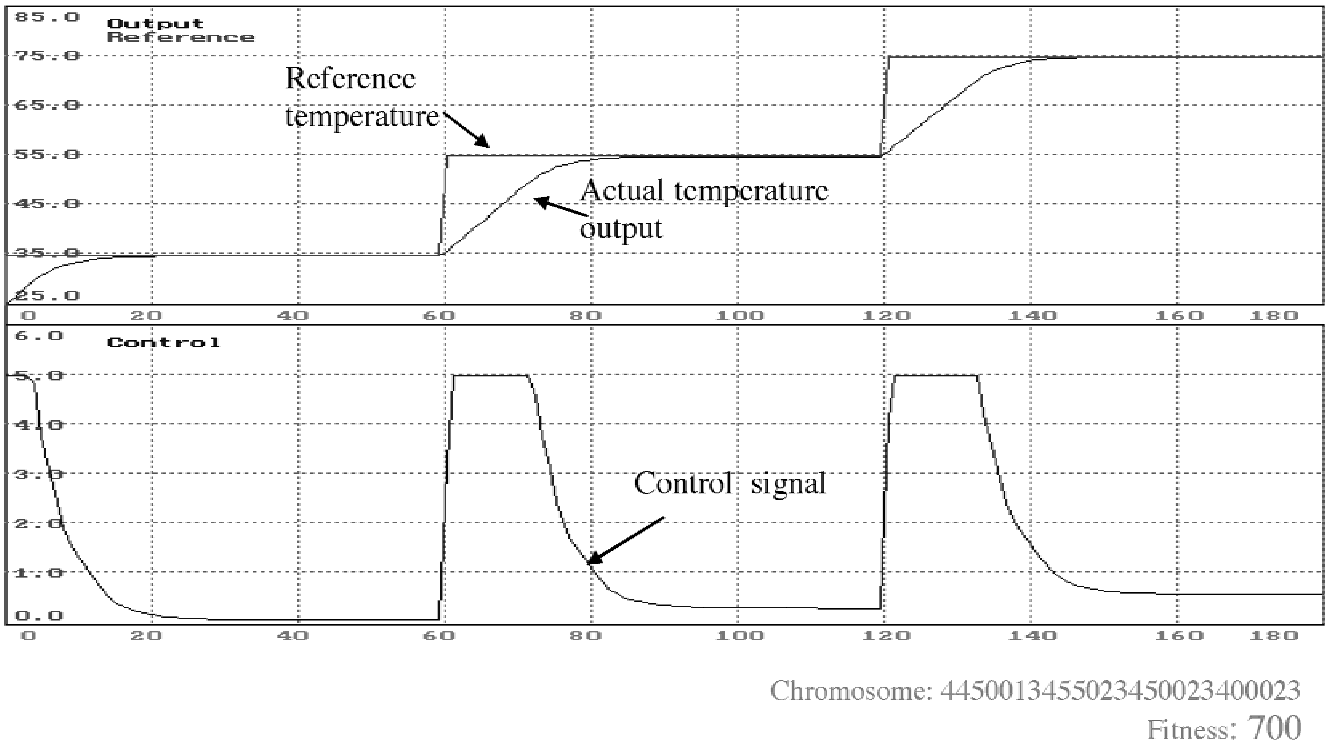
The input and output variables are described by the linguistic terms set {NL, ZE, PL, PM, PH} and {HD, LD, NC, LI, HI}, respectively. The genetic codes for the linguistic terms are 1, 2, 3, 4, and 5. With five labels, the maximum number of rules is 25, which is also the length of the chromosome string. The number of rules in a chromosome during the initial search is set to 13.

In this paper, we let IGALM learn the efficient rule set. Earlier, in Figure 1, the rules were based on preconceived knowledge and intuition. Here, IGALM is capable of producing functional rules without any prior knowledge of the system's behavior or the guidance of a human expert. The IGALM has demonstrated its capability to find suitable solutions for the domain problem. It starts from a population of randomly created chromosomes and then directs the population to stronger chromosomes through generations of evolution. The final outcome of the IGALM is a 10-rule fuzzy system, and the corresponding linguistic rule matrix is shown in Figure 12. As stated earlier, the desired target temperatures during the training stage are 40, 60, and 80°C. To evaluate the functional performance of the derived fuzzy controller, we changed the temperature set point $r(k)$ to 35, 55, and 75°C. We then compared the performance of a manually designed 17-rule fuzzy system with the one generated

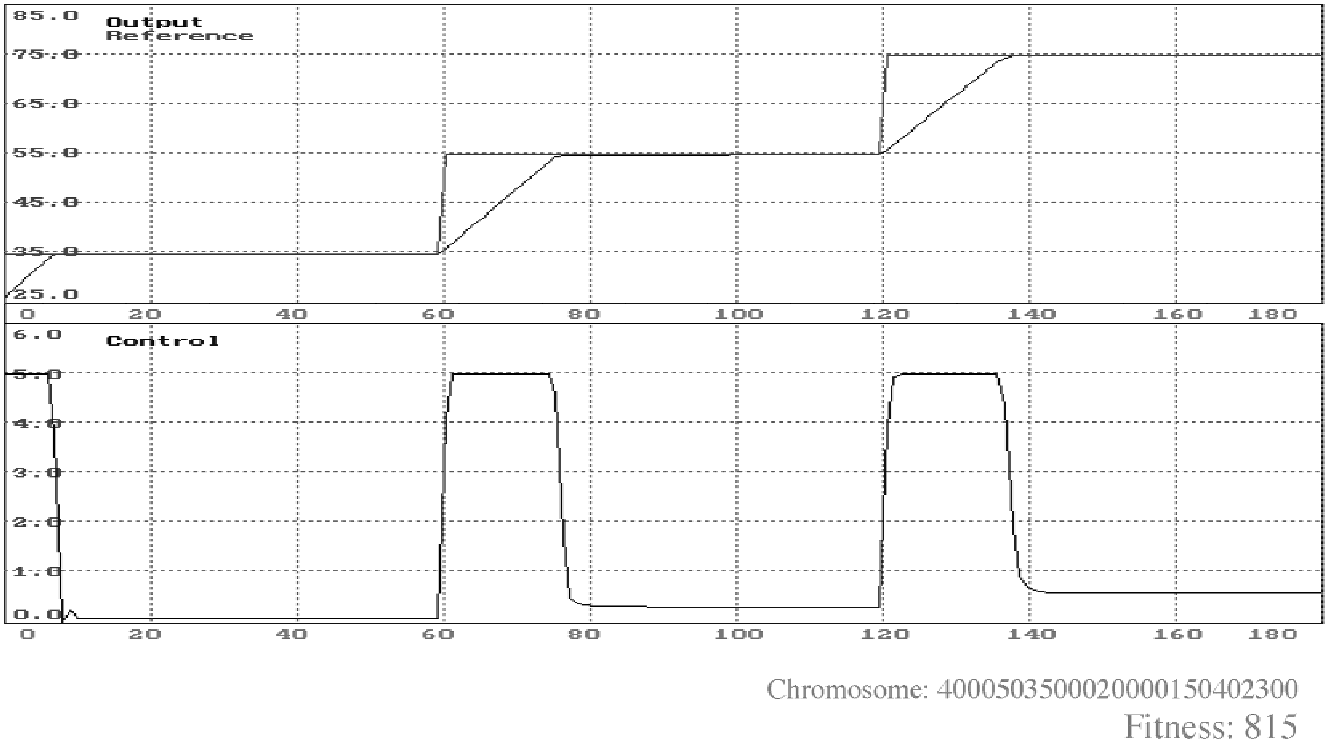
| $\frac{dError}{dt}$ \ Error | NL | ZE | PL | PM | PH |
|---|---|---|---|---|---|
| NL | LI | | | | HI |
| ZE | | NC | HI | | |
| PL | | LD | | | |
| PM | | HD | HI | | LI |
| PH | | LD | NC | | |

**Fig. 12.** A derived 10-rule fuzzy system.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press




(a)




(b)

**Fig. 13.** The simulation results using three temperature settings (35, 55, and 75°C) for (a) a 17-rule and (b) a 10-rule system.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

by the IGALM. The functional performance of these two sets of rules is shown in Figure 13 for the plant simulation run, showing the capability of the control system to maintain the temperature. In Figure 13, the top half of each plot displays the temperature response of the plant and the lower half shows the control action (voltage) taken by the controller in maintaining the temperature at the given set points. In general, both fuzzy systems exhibit desirable performance in controlling the temperature of the water bath. However, the system derived by IGALM, besides being more efficient (smaller number of rules), gives a faster response then the system derived manually.

## 5. CONCLUSIONS

The integration of fuzzy systems and GAs has produced a beneficial outcome, resulting in many useful practical applications. A paradigm for learning fuzzy rules is described and demonstrated on the water bath temperature control problem. Such an integrative approach promises to ease the design of the fuzzy system by automating the formulation of the if–then rules by means of a GA.

In this work we place emphasis on managing the size of the GA solution space so as to improve the GA search efficiency. Instead of using the entire solution space, we systematically partition it into smaller subspaces. The GA then focuses its search for an optimum solution on each of the subspaces. Once the current subspace is conquered, the GA proceeds to uncover a new solution from a new subspace. This way the GA is able to iteratively derive a compact set of *n* rules. Results presented have shown that the rule sets generated by the IGALM perform equally well as or even better than the ones defined by a human expert.

With IGALM, one is able to do away with the need for domain expertise or mathematical analysis of the plant's model. Instead, what is required is to specify the problem domain as a set of objectives to guide the GA search. The GA conducts parallel search for rule sets that achieve the objectives specified. Successive runs of the GA end up with different solutions of varying fitness measures. As a result, the best solution chosen may have some rules different from those created through human intuition and logic. In this way, GA contributes by discovering new solutions and possibly more efficient rule sets that achieve the same desired outcome. Such rule sets may not be conceivable based on human logic and intuition.

For systems that operate in a rapidly changing environment, the advantages accorded by the presence of a GA in producing an adaptive fuzzy system are significant. We should continue to develop better models of integrating fuzzy systems and GAs that are applicable towards real-life practical situations. Finally, controllers for other problems should be examined to further show the effectiveness of this method.

## REFERENCES


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

Goldberg, D.E., & Lingle, R. (1985). Alleles, loci and the travelling salesman problem. *Proc. First Int. Conf. Genetic Algorithms and Their Applications*, pp. 154–158.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*, pp. 76–79. New York: Addison–Wesley.

Hoffmann, F., & Pfister G. (1997). Evolutionary Design of a Fuzzy Knowledge Base for a Mobile Robot. *International Journal of Approximate Reasoning, 17(4)*, 447–469.

Homaifar, A., & McCormick, E. (1995). Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms. *IEEE Transactions on Fuzzy Systems*, *3*, 129–139.

Juang, C.F., Lin, J.Y., & Lin, C.T. (2000). Genetic Reinforcement Learning through Symbiotic Evolution for Fuzzy Controller Design. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, *30(2)*, 290–301.

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

Khalid, M., & Omatu, S. (1992) A neural network controller for a temperature control system. *IEEE Control Systems Magazine*, *12*, 58–64.

Lim, M.H., Rahardja, S., & Gwee, B.H. (1996). A GA paradigm for learning fuzzy rules. *Fuzzy Sets and Systems*, *82*, 177–186.

Lim, M.H., Yuan, Y., & Omatu, S. (2000). Efficient genetic algorithms using simple genes exchange local search policy for the quadratic assignment problem. *Computational Optimisation and Applications*, *15(3)*, 249–268.

Lin, C.T., Juang, C.F., & Li, C.P. (1999). Temperature control with a neural fuzzy inference network. *IEEE Transactions on Systems, Man, and Cybernetics*, *29*, 440–451.

Magdalena, L. (1998). Crossing unordered sets of rules in evolutionary fuzzy controllers. *International Journal of Intelligent Systems, 13(10–11)*, 993–1010.

Ng, K.C., & Li, Y. (1994). Design of sophisticated fuzzy logic controllers using genetic algorithms. *Proc. Third IEEE Int. Conf. Fuzzy Systems*, *3*, 1708–1712.

Ng, W.L., & Lim, M.H. (2002). Genetic optimisation of fuzzy rule set for industrial plant automation. *Fourth Asia–Pacific Conf. Simulated Evolution and Learning*, pp. 627–631.

Tanomaru, J., & Omatu, S. (1992). Process control by on-line trained neural controllers. *IEEE Transactions on Industrial Electronics*, *39*, 511–521.

Thrift, P. (1991). Fuzzy logic synthesis with genetic algorithms. *Proc. Fourth Int. Conf. Genetic Algorithms*, pp. 509–513.


---

**Meng-Hiot Lim** has been a faculty member of the School of EEE, Nanyang Technological University (NTU), since 1989. He received his BS, MS, and PhD from the University of South Carolina. At present, he is an Associate Professor, with research interests in the area of computational intelligence, e-based applications, reconfigurable circuits and architecture, and financial applications of graph theory. Dr. Lim has held positions as Reliability Engineer at International Rectifier in California and Visiting Associate Professor in the Department of ECE, University of Missouri–Rolla. Currently, he is also on concurrent appointment as Deputy Director of the Center for Financial Engineering, a multidisciplinary research center anchored at the Nanyang Business School. The Centre, besides promoting multidisciplinary research, oversees the MS in Financial Engineering program, where he played a significant role in the planning and formulation of the curriculum. During his time

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press

with NTU, he has also offered technical consultancy for companies and delivered specialized short courses. Dr. Lim has recently been playing an active role in nurturing and supporting an Internet startup to commercialize proprietary intellectual property, both as a principal investigator of collaborative R&D projects and as advisor to the company. Dr. Lim is a regular participant and organizer of major international conferences and publishes regularly in technical journals. He was recently the Local Arrangement Chair of ICONIP/SEAL/FSKD-2002 held in Singapore and served as Asian Liaison for IJCNN-03 (Portland, OR). He is a firm believer of an educational paradigm that veers toward empowering students to explore and challenge norms beyond classroom reality.

**Willie Ng** received his BEng (Hon) degree in electronics and electrical engineering from the University of Edinburgh in 2001. He is currently a postgraduate in the School of EEE, Nanyang Technological University. He is also a part-time Lecturer in the Institute of Technical Education. His research interest is in computational intelligence involving neural networks, evolutionary programming, and fuzzy computing.

https://doi.org/10.1017/S0890060403174057 Published online by Cambridge University Press