A Simulated Evolution Algorithm for The Synthesis of Trapezoid-Shaped Membership Functions

梯形歸屬函數合成之模擬進化演算法

Shih-Hsu Huang
黃世旭
Department of Electronic Engineering, Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.
中原大學電子工程學系
桃園縣中壢市普忠里
普仁 22 號
shhuang@cycu.edu.tw

Wen-Hon Peng
彭文宏
Department of Electronic Engineering, Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.
中原大學電子工程學系
桃園縣中壢市普忠里
普仁 22 號
g8976036@ cycu.edu.tw

Jian-Yuan Lai
賴建元
Department of Electronic Engineering, Chung Yuan Christian University, Chung Li, Taiwan, R.O.C.
中原大學電子工程學系
桃園縣中壢市普忠里
普仁 22 號
g8976039@cycu.edu.tw

Abstract

In the past few years, there have been an increasing number of systems based on fuzzy logic in various fields of applications, such as ATM network control, GPS data processing, aircraft flight control, image processing, and so on. In a fuzzy system, the fuzzy rules are characterized by membership functions. Therefore, when designing a fuzzy system, the membership functions play a very important role because they are the "brain" in the fuzzy inference process. In order to better control the fuzzy system, we need to have an effective design methodology to synthesize the membership functions. In this paper, we will present a simulated evolution based algorithm for automatic synthesis of the trapezoid-shaped membership functions, which are based on the representation format presented in [1,2]. Given some test patterns and their desired outputs, the proposed approach iteratively improves the membership functions by simulated evolution. Our optimization goal is to find suitable trapezoid-shaped membership functions as the knowledge base such that the total control error is minimized. The main distinction of the proposed approach is that it fully utilizes the properties inherited in trapezoid-shaped membership functions. As a result, the possibilities of variations in the simulated evolution can be accurately estimated. Experimental data shows that the proposed approach achieves very good results and thus is well suitable for the fuzzy logic controller presented in [1].

Keywords: Fuzzy System, Membership Functions, Simulated Evolution Algorithm, Control Error, Automatic Synthesis, Fuzzy Logic, and Fuzzy Logic Controller.

摘要

近幾年，模糊邏輯已在不同應用領域，有著極為廣泛的運用；例如，網路流量控制、衛星定位系統訊號處理、航空飛行控制、影像處理等，都有運用模糊系統實現其應用之例子。在一個模糊系統中，主要是以歸屬函數特徵化其模糊規則。因此，歸屬函數在模糊歸論時，扮演類似大腦思考的角色。所以，歸屬函數的設計，對於模糊系統之建構，極為重要。為了能有效的進行模糊控制，我們必須有一套設計方法可以進行歸屬函數的合成。本篇論文，我們將提出一個以模擬進化演算法為基礎的梯形歸屬函數自動合成系統。此梯形歸屬函數的表示方法，已發表於 [2]， 而可處理此表示方法的模糊邏輯控制器，亦已發表於 [1]。本篇論文，便是以我們之前的研究為基礎，提出一個梯形歸屬函數自動合成系統，以便與 [1,2] 之硬體研究成果相整合，建構一個完整的模糊系統設計環境。本篇論文所提出之梯形歸屬函數自動合成系統，其系統輸入主要是由使用者提供部份已知之測試向量及其需求結果，藉由模擬進化演算法，反覆調整歸屬函數，以改善原先歸屬函數之設計。我們最佳化的目標，是將歸屬函數調整到最適當位置，以達到最小之系統控制誤差。我們所提出的演算法，主

要的特色，是充分利用梯形歸屬函數的特性，得以準確推估歸屬函數各種調整變異方式之機率。實驗結果顯示，我們的演算法可以得到很小之控制誤差，因此極為適用於以[1] 所提出之模糊邏輯控制器為基礎之模糊系統設計。

關鍵字：模糊系統、歸屬函數、模擬進化演算法、控制誤差、自動合成、模糊邏輯、模糊邏輯控制器。

1. Introduction

During the past two decades, fuzzy logic control has emerged to be one of the most active and successful areas in scientific researches and industrial applications. By the efforts of countless researchers, there have been an increasing number of applications based on fuzzy logic in various fields of applications, such as ATM network control [3], GPS data processing [4], decision-making support system [5], automobile transmission control, automatic train operation systems [6], image processing [7,8], process control [9,10], and signal processing [11]. The reasons for the success of fuzzy logic lies in its capacity to tolerate information expressed in a way, which is uncertain, imprecise, and at times only qualitative. Problems, which are too complex or even impossible to deal with using traditional quantitative techniques, can often be solved with a limited computational complexity when a fuzzy approach is used.

The underlying idea of fuzzy approach is to build a model of human expert who is capable of controlling the plant without thinking in terms of a mathematical model. The fuzzy rules and fuzzy logic controller are the two important parts for modeling a fuzzy system. The fuzzy rules define the knowledge base of the control expert, while fuzzy logic controller performs the fuzzy inference.

In a fuzzy system, the fuzzy rules are characterized by membership functions and stored in the memory. A great number of fuzzy systems use general-purpose processors to perform the function of fuzzy logic controller. But dedicated hardware implementation is necessary for real-time fuzzy applications. The digital hardware fuzzy logic controller originated from Togai and Watanabe [12]. Many variations [13,14] have been proposed to improve the inference performance.

In [1,2], we have proposed a high-speed VLSI fuzzy logic controller, which is well suitable for real-time fuzzy applications. By using 0.35μm process as the target technology, the speed reaches up to 3.75M FLIPS (Fuzzy Logic Inferences Per Second). The speedup is achieved by an effective architecture for the computations of trapezoid-shaped membership functions. As a result, the format of membership function is limited to trapezoid shape.

When designing a fuzzy system, membership functions play a very crucial role because they are the “brain”. In order to reduce the control error, the membership functions should be constructed carefully. The construction of membership functions can be intuitive or be based on some algorithmic operations. The optimization goal is to find suitable membership functions as the knowledge base such that the total control errors are minimized.

In order to better control the fuzzy system designed with our high-speed fuzzy logic controller, we need to have an effective approach to synthesize trapezoid-shaped membership functions. In this paper, we will present a simulated evolution algorithm to tackle this problem. The main distinctions of the proposed approach are elaborated as the below:

- It is specific to trapezoid-shaped membership functions. Therefore, it is well suitable for our fuzzy logic controller.
- More importantly, it fully utilizes the properties inherited in trapezoid-shaped membership functions. As a result, the convergence can be easily achieved and hence the CPU time can be saved.

The remainder of the paper is organized as follows. Section 2 will survey the related concepts, especially the trapezoid-shaped membership functions for our VLSI fuzzy logic controller. Then, in Section 3, we will present the formulations for the problem of synthesizing membership functions. The proposed simulated evolution algorithm will be described in Section 4. Experimental results and the comparisons will be shown in Section 5. Finally, some concluding remarks will be given in Section 6.

2. Preliminaries

In this section, we will provide an overview of the related works. Firstly, in Section 2.1, we will brief the concept of fuzzy logic. Then, in Section 2.2, we will introduce the function of fuzzy inference. Finally, in Section 2.3, we will review our format for trapezoid-shaped membership functions and our VLSI architecture to tackle the fuzzy inference in terms of the format.

2.1 Fuzzy Logic

Zadeh proposed the extension of ordinary set theory to fuzzy sets [15]. Fuzzy logic replaces "true" and "false" with continuous set membership values ranging from ZERO to ONE, which mirrors natural language concepts. Therefore, a fuzzy set, in fact, is a membership function $\mu_s : M \rightarrow [0,1]$, which associates each element in the universe disclosure set M with a real value between ZERO and ONE. For example, action is required when "the speed is approximately 30 km/h", as shown in Figure 1, so 25 km/h returns a degree of membership of 0.7. In other words, the membership function μ_s can be defined as a fuzzy set $S = \{(m_i, \mu_s(m_i)) \mid m_i \in M\}$, where $0 \leqq \mu_S(m_i) \leqq 1$.

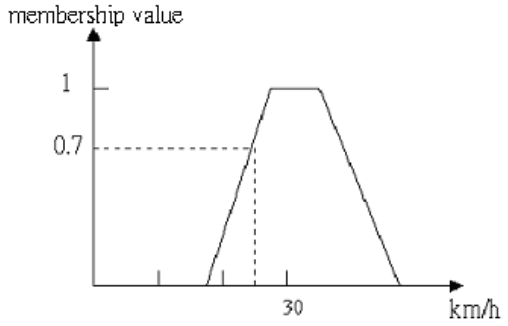

Figure 1. An example used to illustrate the concept of membership function.

2.2 Fuzzy Inference Process

Since the inputs from a plant are crisp values, a fuzzification operator *Fuzz* is used first. Fuzzification could be defined as a mapping from an observed input space to membership functions in certain input universes of disclosure.

Without loss of generality, fuzzy logic controller usually suppose that the fuzzy system has two-input (X and Y) and one-output (O) with r fuzzy rules of the form:

Rule R_i: IF (X is A_i) and (Y is B_i) then (O is C_i)

where A_i and B_i, are the antecedent membership functions associated with the linguistic (fuzzified) input variables X and Y, respectively, and C_i is the consequent membership function associated with the linguistic (fuzzified) output variable O.

The fuzzy inference process can be depicted as the below. Firstly, fuzzified inputs X and Y are simultaneously broadcasted to all r fuzzy rules to be compared with the antecedent parts. The matching degrees between (X and A_i) and between (Y and B_i) are given by the max-min calculation method as the following two equations:

$$\alpha_i^A = \max_m(\min_m(X(m), A_i(m)))$$

$$\alpha_i^B = \max_m(\min_m(Y(m), B_i(m)))$$

The weight of rule i is calculated by:

$$\omega_i = \min(\alpha_i^A, \alpha_i^B)$$

Hence, rule R_i recommends a control decision as follows:

$$O_i(m) = \min_m(\omega_i, C_i(m))$$

Last, the combined fuzzy results of all control rules is given by:

$$O(m) = \cup_i O_i(m) = \max(O_0(m), ..., O_{r-1}(m))$$

Since the inference process should output some crisp control results practically, it necessitates the use of a defuzzifier. If the center of gravity (COG) algorithm is employed, the final crisp output o can be calculated as:

$$o = \frac{\sum_m (m \times O(m))}{\sum_m O(m)} \qquad (1)$$

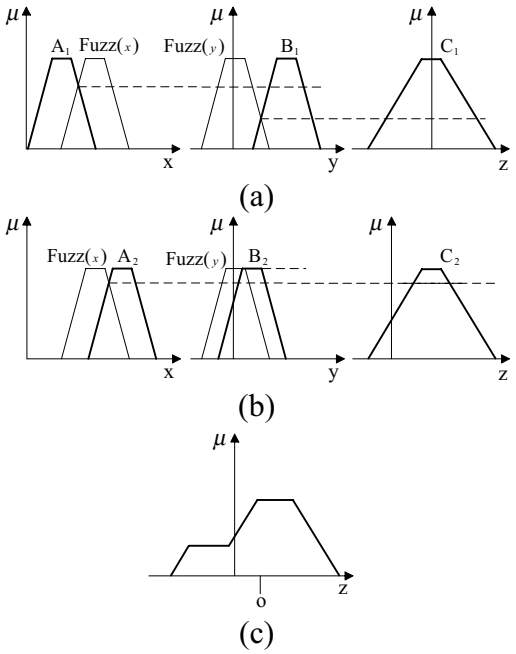

Figure 2. A graphical implication for max-min calculation method with fuzzified inputs.

Let us use Figure 2 as an example. Suppose that two fuzzy rules R1 and R2 are designed in the system. The antecedent membership functions A_1 and B_1 and the consequent membership function C_1 are

associated with rule R_1. Figure 2 (a) shows the fuzzy inference process. The antecedent membership functions A_2 and B_2 and the consequent membership function C_2 are associated with rule R_2. Figure 2 (b) shows the fuzzy inference process. By combining the results of rule R_1 and rule R_2 and applying the COG computation, the crisp output o is obtained as shown in Figure 2 (c).

2.3 Our Fuzzy Logic Controller

In real-time applications, it is therefore necessary to use hardware architecture dedicated to fuzzy computation. In [1], we have proposed a high-speed VLSI fuzzy logic controller to increase inference speed. Our basic idea is to propose an effective format for trapezoid-shaped membership functions. Then, by making use of both the advantage of the new format and the property inherited in trapezoid shape, the latency of a fuzzy inference can be considerably reduced. Furthermore, in order to exploit temporal parallelism, the activity of fuzzy inference is divided into four pipelining stage. The proposed architecture has been implemented by using the design kit of CIC 0.35μm cell library and the processing speed reaches to 3.75M FLIPS. The detailed pipelined architecture is given in [1].

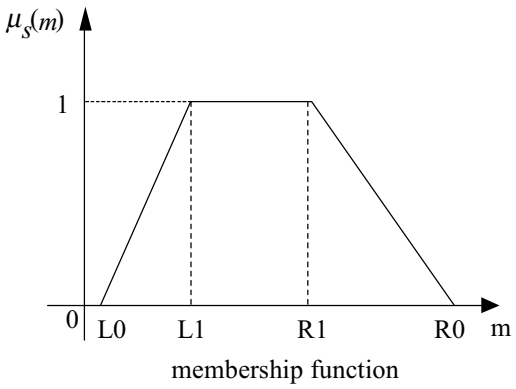

Figure 3. An example used to illustrate our format.

In order to represent all possible trapezoid shapes for membership functions, our format is to store the four corner points. As a result, our new format uses four tuples (L0, L1, R1, R0) to describe a trapezoid-shaped membership function, in which L0 is the leftmost x-axis point in the lower side of the trapezoid, L1 is the leftmost x-axis point in the upper side of the trapezoid, R1 is the rightmost x-axis point in the upper side of the trapezoid, R0 is the rightmost x-axis point in the lower side of the trapezoid. Figure 3 gives an example to illustrate the format. As shown in [2], the new format may achieve higher inference speed with fewer memory requirements.

3. Problem Definition

For real-time fuzzy logic control, in addition to the speed, the accuracy is also very important. In order to ensure the accuracy, we need an effective methodology to derive the fuzzy inference rules. Because the fuzzy inference rules are represented by membership functions, the problem we study in this paper is: Given some test patterns with their desired outputs, to modify the associated membership functions to more appropriate shapes. The constraint is that all the membership functions are limited to trapezoid shapes. The optimization goal is to minimize the total control error of test patterns.

3.1 The Formulations

Suppose that the fuzzy system has r fuzzy inference rules. For each rule R_i, it is described by antecedent membership functions A_i and B_i, and consequent membership function C_i, where $i = 1, 2, \ldots,$ and r. Initially, all the membership functions are specified by the system designer.

Assume that we are given k pairs of test patterns to verify the control performance of the specified membership functions. The test patterns are (x_1,y_1), (x_2,y_2), …, and (x_k,y_k), and their desired outputs are o_1, o_2, …, and o_k, respectively. However, based on the specified membership functions, the outputs of these test patterns are o_1^*, o_2^*, …, and o_k^*, respectively. We say that the difference between o_i and o_i^* is the control error of the test pattern (x_i,y_i), where $i = 1, 2, \ldots,$ and k.

As described in Section 2.3, we use four tuples to represent a trapezoid-shaped membership function. Suppose that M is a membership function in our format. For the convenience of presentation, we denote M^1 is the first element, M^2 is the second element, M^3 is the third element, and M^4 is the fourth element, respectively. It is obvious that $M^1 \leqq M^2 \leqq M^3 \leqq M^4$.

Our goal is to design an automatic synthesis system to construct the membership functions in appropriate places. The inputs are r fuzzy inference rules and k test patterns with their desired outputs. The cost function is to minimize

$$\sum_{i=1}^{k}(o_i - o_i^*)^2 \qquad (2)$$

Subject to:

- $|(o_i - o_i^*)| \le e$, where i = 1, 2, …, and k, and *e* is the specified error tolerance.
- $A_i^1 \le A_i^2 \le A_i^3 \le A_i^4$, where i = 1, 2, …, and r.
- $B_i^1 \le B_i^2 \le B_i^3 \le B_i^4$, where i = 1, 2, …, and r.
- $C_i^1 \le C_i^2 \le C_i^3 \le C_i^4$, where i = 1, 2, …, and r.

3.2 Motivation

Firstly, the system designer specify all the membership functions, including A_1, B_1, C_1, A_2, B_2, C_2, …, A_r, B_r, and C_r, by his intuition. Then, based on the initial specified membership functions, our automatic synthesis tool iteratively improves the membership functions by relocating them to appropriate places. As a result, the control error can be minimized.

Let us use Figure 4 as an example. Suppose that Figure 4 (a) is the fuzzy inference process based on initial membership functions, i.e., A_1, B_1, A_2, and B_2. The crisp output o_1^* is obtained as shown in Figure 4 (b). We find that o_1^* is far away from o_1. To reduce the control error, we relocate the places of membership functions B_1 and A_2. The modified fuzzy membership functions and the fuzzy inference process are shown in Figure 5 (a). The crisp output $o_1^{*'}$ is given in Figure 5 (b). We find that the new $o_1^{*'}$ is closer to o_1.

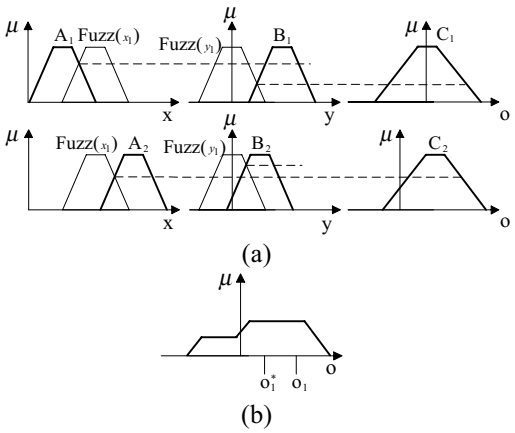

Figure 4. An example. It is used to illustrate the effects of antecedent membership functions on fuzzy inference.

Our basic idea is to iteratively improve the membership functions by simulated evolution. For a fuzzy rule R_i, it is characterized by three membership functions A_i, B_i, and C_i. Therefore, we can use the three membership functions A_i, B_i, and C_i as the genes of the membership functions R_i. As a result, for a fuzzy system with *r* fuzzy inference rules, each generation of the simulated evolution has *3r* genes. We define that the variation of gene is the relocation of membership function. Then, our problem is to derive reasonable probabilities of variations such that the best generation can be achieved quickly.

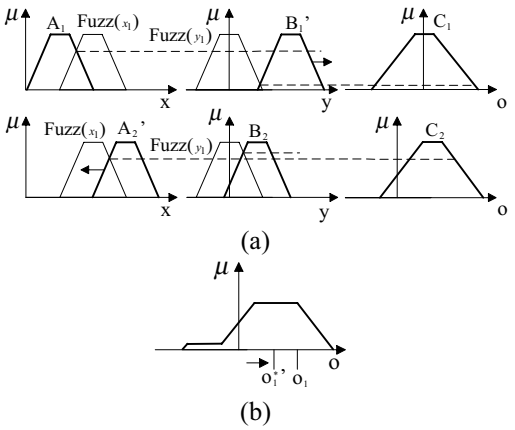

Figure 5. The example shows the result of fuzzy inference after the places of membership functions B_1 and A_2 are relocated.

4. The Proposed Algorithm

Firstly, the system designer specifies the trapezoid-shaped membership functions by his intuition. In many cases, the membership functions can be further improved. In the following, we will present a simulated evolution based algorithm to optimize the membership functions. Our optimization goal is to minimize the total control error under the constraint of trapezoid shape and the specified error tolerance of each test pattern. Figure 6 gives our simulated evolution algorithm.

The first generation of the simulated evolution is the initial fuzzy system design. We define the membership functions are the genes of the evolution. A possible variation of a gene is the relocation of membership function. The simulated evolution algorithm will iterate many generations until no better result can be obtained within n generations, where n is the parameter specified by the user. The best result G_{best} among all the generations is returned as the output.

```
Procedure our_approach( );
begin
  count=0;
  repeat
  G_new=find_a_new_generation( );
    if (cost(G_new) < cost(G_current)) then
      begin
        G_best= G_new;
        G_current= G_new;
        count=0;
      end
    else
      begin
        generate a random number x ;
        if ( x <e^(-αt) ) then
          G_current= G_new;
        else
          count=count+1;
      end
  until (count > n);
end
```

Figure 6. The algorithm of our simulated evolution approach

According to the current generation $G_{current}$, the routine find_a_new_generation produces a new generation G_{new} with some variations of genes. Whenever the new generation causes lower cost, the new generation is accepted. Otherwise, the new generation is accepted with a probability $e^{-\alpha t}$, where is a fixed positive parameter and t is the number of iterations. If a new generation is accepted, it becomes $G_{current}$. Otherwise, the $G_{current}$ remains no change.

For the evolution of a new generation, each gene may have a variation. Figure 7 gives the pseudo code of the routine find_a_new_generation. There are two important tasks in this routine. At first, it should determine the probabilities of all the possible variations. This step is very important because a good criterion would assist faster design convergence. Next, it varies each gene based on the specified probabilities.

```
Procedure find_a_new_generation( );
begin
  build_the_probability_table( );
    for i=1 to r do
      begin
        generate a random number x;
          if (x > η) then
            generate_a_variation(C_i);
          else
            begin
              generate_a_variation(A_i);
              generate_a_variation(B_i);
            end
      end
  end
```

Figure 7. The pseudo code used to find a new generation.

During the inference of a fuzzy rule, both the antecedent membership functions and consequent membership function are the factors to final crisp output. To reduce the problem complexity, we prefer to fix one factor and change another factor at one time. The probability to change which factor is determined by the probability η, which is specified by the user.

```
Procedure generate_a_variation(M);
begin
  for P =1 to 4 do
    begin
      generate a random number x;
        if (x < Prob(M(left))) then
          M^p=min(M^(p-1),M^p-1);
        else if (x<Prob(M(left))+Prob(M(right)))
          M^p= M^p+1;
        end
    end
```

Figure 8. The pseudo code used to generate a variation of a membership function M.

In our architecture, we use four corner points to characterize a membership function. Therefore, the variation of a gene also means the variations of the four corner points. Figure 8 gives the pseudo code. Given a membership function M, the routine generate_a_variation tries to modify the corner points M^1, M^2, M^3, and M^4, respectively. We assume that each corner point has the same probability distribution of position variation. Let Prob(M(left)) and Prob(M(right)) denote the probabilities of left shift and right shift of the membership function M, respectively. Although probability distributions are the same, individual random numbers are used to vary the corner points, respectively. For each corner point M^P, our methodology is as the below. Suppose that the condition of left shift happens, M^P minus 1 if $M^{P-1} < M^P$. On the other hand, if the condition of right shift happens, the value of corner point plus 1.

An accurate estimation of the probabilities of variations would greatly speedup the evolution. Given a test pattern, suppose that the obtained crisp output o_i^* is greater than the desired output o_i, we have the following two methods to improve the system:

(1) The first method is to shift left the consequent membership function. As a result, o_i^* is decreased.

(2) The second method is to decrease the weights of rules. For each rule R_j, where j = 1, 2, …, and r, the procedures are elaborated as the below. Firstly, we determine which antecedent membership function contributes to the weight w_j by comparing the matching degrees α_j^A and

α_j^B. Suppose that A_j decide the weight w_j. The method to decrease the weight w_j is as the below. If A_j is left to the fuzzified input, we should shift A_j left. If A_j is right to the fuzzified input, we should shift A_j right. Figure 9 gives an example.

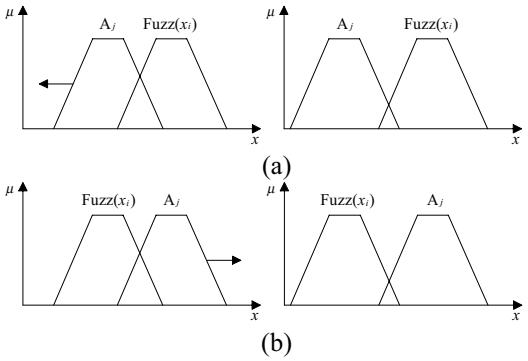

Figure 9. (a) The condition of A_j is left to the fuzzified input; (b) the condition of A_j is right to the fuzzified input

Similarly, suppose that the obtained crisp output o_i^* is less than the desired output o_i, we have the following two methods to improve the system:

(1) The first method is to shift right the consequent membership function. As a result, o_i^* is increased.

(2) The second method is to increase the weights of rules. For each rule R_j, where j = 1, 2, …, and r, the procedures are elaborated as the below. Firstly, we determine which antecedent membership function contributes to the weight w_j by comparing the matching degrees α_j^A and α_j^B. Suppose that A_j decide the weight w_j. The method to increase the weight w_j is as the below. If A_j is left to the fuzzified input, we should shift A_j right. If A_j is right to the fuzzified input, we should shift A_j left.

For each membership function M, we define two counters: M(left) and M(right). A fuzzy rule is associated with six counters, including A_j(left), A_j(right), B_j(left), B_j(right), C_j(left), and C_j(right). Whenever we want to shift left membership function M, the counter M(left) plus 1. On the other hand, whenever we want to shift right membership function M, the counter M(right) plus 1. Note that, initially, the values of all the counters are zero.

The probabilities Prob(M(left)) and Prob(M(right)) are defined as M(left)/k and M(right)/k, respectively, where k is the number of test patterns.

5. Experimental Results

The proposed simulated evolution algorithm has been implemented in a C program running on an Intel Pentium-III personal computer. Two benchmarks are used to test the effectiveness of the synthesis methodology. The first test case is a truck backer-upper control system, which is adopted from [16]. The second test case is nonlinear control example, which is adopted from [17]. Both experimental data shows that our approach can achieve very good results. The details of our experiments are elaborated as the below.

5.1 Truck Backer-Upper Control System

The truck backer-upper control system is designed with the goal to park the truck in a prescribed parking lot. Three variables x, y and θ were defined to describe this system, where the variable θ specifies the angle of the truck to the horizontal while the coordinate pair (x,y) specifies the position of the rear center of the truck in the plane [0,100]x[0,100]. The details including the mathematical relationships among these variables are described in [16]. Table 1 shows the 5x7 rule table. Figure 10 shows the diagram of simulated truck and Figure 11 gives the fuzzy membership functions for x, ϕ and θ.

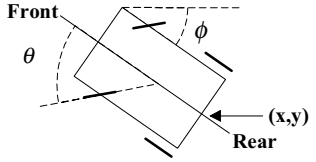

Figure 10. Diagram of simulated truck.

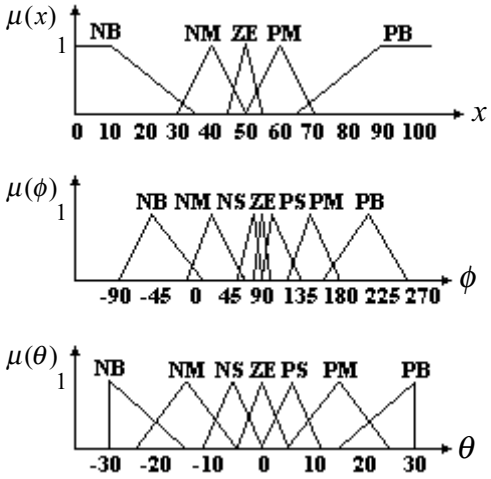

Figure 11. Fuzzy membership functions for x, ϕ and θ with 5x7 rules.

Table 1. The 5×7 rule table of truck backer-upper control system.

B \ A	NB	NM	ZE	PM	PB
NB	PS	PM	PM	PB	PB
NM	NS	PS	PM	PB	PB
NS	NM	NS	PS	PM	PB
ZE	NM	NM	ZE	PM	PM
PS	NB	NM	NS	PS	PM
PM	NB	NB	NM	NS	PS
PB	NB	NB	NM	NM	NS

In this experiment, we try to implement the fuzzy system using our fuzzy logic controller. The membership function shown in Figure 11 cannot be used directly, because the size of universe disclosure set of our fuzzy logic controller is 64. Therefore, we need to map the variables x, ϕ and θ to the interval from 0 to 63. As a result, the membership functions, including antecedent membership functions and consequent membership function, need modifications.

Firstly, we derive 3600 test patterns and their desired outputs. These test patterns are used to train the simulated evolution algorithm. The variables x and ϕ are associated with antecedent membership functions A and B, respectively, and the variable θ is associated with consequent membership function C. Initially, we assume that the membership functions are uniformly distributed in the space of universe disclosure set with triangular shapes. Note that triangular shape is a special case of trapezoid shape.

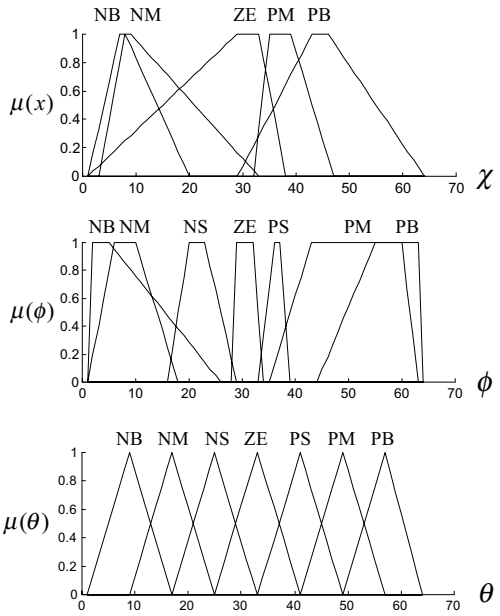

Figure 12. The membership functions generated by our algorithm.

With 11787 generations in the evolution the result is obtained. Figure 12 gives the synthesized membership functions. We can see that they cover all entire range. The average control error is 1.88. Figure 13 shows the original three-dimensional control surface according to data pairs given in [16]. Using the same test patterns and the synthesized membership functions, a very similar control surface can be obtained as shown in Figure 14. Therefore, our approach achieves very good results. This result is listed in the case 1 of Table 2.

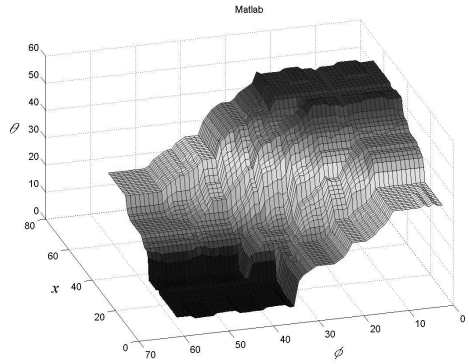

Figure 13. The control surface of truck backer-upper control system using the data pairs given in [16].

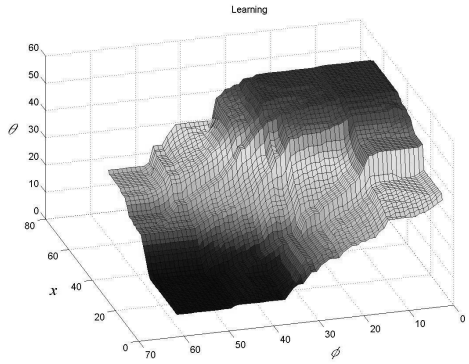

Figure 14. The control surface generated by the synthesized membership functions, which are shown in Figure 12.

Another concern is that the effects of initial membership functions in the simulated evolution. Table 2 tabulates the experimental results for six cases of different initial membership functions. The first column gives the initial average control error in average. The last three columns presents the synthesized results, including the final average control error, the total number of generations, and the CPU

time. Experimental data shows that our approach can converge to a small value no matter the given initial membership functions.

Table 2. Our experimental results for the example of truck backer-upper control system.

Case	Initial average control error	Results		
		Final average control error	Total number of generations	CPU run time (second)
1	4.87	1.88	11787	9092
2	5.27	1.94	19460	12418
3	5.67	1.82	4246	2686
4	5.44	1.99	16539	10983
5	5.57	1.92	4662	3596
6	5.74	1.92	11852	7530

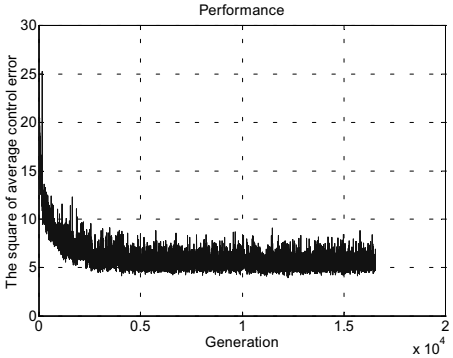

Figure 15. Average cost value of simulated evolution algorithm at each generation of truck backer-upper control system.

5.2 Nonlinear Control

The nonlinear example to be controlled is represented by

$$O = (2x+4y^2+0.1)^2 \qquad (3)$$

The system has two inputs and one output. There are 441 pairs of source input-output values collected as the test patterns. In order to let the test patterns to fit our presented fuzzy logic controller, we map variables x, y and o to the initial from 0 to 63. With the 441 training test patterns and initial membership functions, our algorithm generates appropriated control rules. After 12717 generations evolute, the final result is obtained. The control surface generated by the synthesized membership functions is shown in Figure 16, which is almost the same as the control surface given in [17]. This result is listed in the case 1 of Table 3. Note that the control errors also exist in the fuzzy system of [17]. The control errors are caused by hardware limitations, e.g., the precision of floating points.

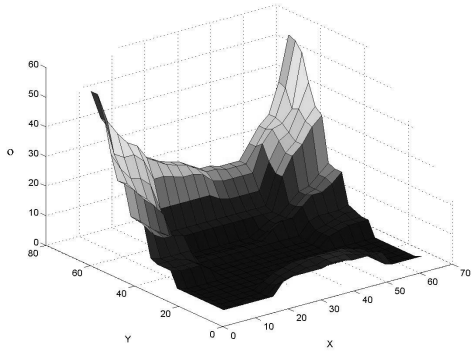

Figure 16. The control surface generated by the synthesized membership functions.

Table 3 tabulates the experimental results for six different cases of initial membership functions. The first column gives the initial average control error in average. The last three columns presents the synthesized results, including the final average control error, the total number of generations, and the CPU time. Experimental data shows that our approach can converge to a small value no matter the given initial membership functions.

Table 3. The experimental results for the example of nonlinear control.

Case	Initial average control error	Results		
		Final average control error	Total number of generations	CPU run time (second)
1	3.91	1.94	12717	1862
2	3.62	1.97	11413	1679
3	2.42	1.72	19889	2926
4	2.28	1.70	8731	1285
5	3.54	1.90	15629	2204
6	4.07	1.95	17515	2470

6. Conclusions

In this paper, we presented an automatic synthesis system for the design of trapezoid-shaped membership functions, which are based on the format presented on [1,2]. Based on some test patterns and their desired outputs, our approach is to iteratively improve the membership functions by simulated evolution. The optimization goal is to minimize the total control error. Experimental data shows that the proposed approach achieved very good

result and thus is well suitable for the design of membership functions of fuzzy system that uses the fuzzy logic controller presented in [1].

References

[1] S.H. Huang and J.Y. Lai, "A High-Speed VLSI Fuzzy Logic Controller with Pipeline Architecture", accepted by 10th IEEE International Conference on Fuzzy Systems, 2001.

[2] S.H. Huang and J.Y. Lai, "An Efficient Membership Function Representation for High-Resolution Fuzzy Systems", in the Proc. of the Workshop on the 21st Century of Digital Life and Internet Technologies (CD-ROM), 2001.

[3] G. Ascia, V. Catania, D. Panno, F. Ficili, and S. Palazzo, "A VLSI Fuzzy Expert System for Real-Time Traffic Control in ATM Networks", IEEE Trans. on Fuzzy Systems, vol. 5, pp. 20--31, 1997.

[4] J. L. Chang, Y. Y. Chen, F. R. Hang, "Fuzzy Processing on GPS Data to Improve the Position Accuracy", in the Proc. of Intelligent Systems and Information Fuzzy System Symposium, pp. 557--562, 1996.

[5] Simutis, R.," Fuzzy logic based stock trading system", Proc. of the IEEE/IAFE/INFORMS Conference on Computational Intelligence for Financial Engineering, pp. 19--21, 2000.

[6] S. Yasunobu and S. Miyamoto, "Automatic Train Operation by Predictive Fuzzy Control", in Industrial Application of Fuzzy Control, M. Sugeno, Ed. Amsterdam, The Netherlands: North-Holland, pp. 1--18, 1985.

[7] D. Sinha, P. Sinha, E. R. Dougherty and S. Batmen, "Design and Analysis on Fuzzy Morphological Algorithms for Image Processing", IEEE Trans. on Fuzzy Systems, pp. 570--584, 1997.

[8] Barra, V. and Boire, J.Y., "Automatic segmentation of subcortical brain structures in MR images using information fusion", IEEE Trans. on Medical Imaging, Vol. 20, pp. 549--558, 2001.

[9] L. I. Larkin, "A Fuzzy Logic Controller for Aircraft Flight Control", Industrial Applications of Fuzzy Control, M. Sugeno. Ed. Amsterdam: North Holland, pp. 87--103, 1985.

[10] Shouping Guan, Han-Xiong Li and Tso, S.K, "Multivariable Fuzzy Supervisory Control for The Laminar Cooling Process of Hot Rolled Slab", IEEE Trans. on Control Systems Technology, Vol. 9, pp. 348--356, 2001.

[11] Kuang-Yow Lian, Chian-Song Chiu, Tung-Sheng Chiang, and Peter Liu, "LMI-Based Fuzzy Chaotic Synchronization and Communications", IEEE Trans. on Fuzzy Systems, Vol. 9, No. 4, pp. 539--553, 2001.

[12] M. Togai and H. Watanabe, "Expert System on a Chip: An Engine for Real-Time Approximate Reasoning", IEEE Expert Magazine, vol. 1, pp. 55-62, 1986.

[13] H. Watanabe, W.D. Dettloff, and K.E. Yount, "A VLSI Fuzzy Logic Controller with Reconfigurable, Cascade Architecture", IEEE J. Solid-State Circuits, vol. 25, pp. 376--381, 1990.

[14] T.C. Chiueh, "Optimization of Fuzzy Logic Inference Architecture", Computer, pp. 67--71, 1992.

[15] L.A. Zadeh, "Fuzzy Sets", Inf. Control, vol. 8, pp. 338--351, 1965.

[16] Bin-Da Liu, Chuen-Yau Chen and Ju-Ying Tsao, "Design of Adaptive Fuzzy Logic Controller Based on Linguistic-Hedge Concepts and Genetic Algorithms", IEEE Trans. on System, man and cybernetics, vol. 31, No. 1, pp. 32--53, Feb. 2001.

[17] Jer Min Jou, Pei-Yin Chen and Sheng-Fu Yang, "An Adaptive Fuzzy Logic Controller: It's VLSI Architecture and Application", IEEE Trans. on VLSI Systems, Vol. 8, No. 1, pp. 52--60, 2000.