진화적 방법을 이용한 퍼지제어기의 자동 생성에 관한 연구

이지형, 이광형[1]

A Study on Automatic Generation of Fuzzy Controller by Genetic Algorithm

Jee-Hyong Lee, Hyung Lee-Kwang

요 약

본 논문에서는 주어진 입출력 데이타로부터 유전자 알고리즘을 이용하여 퍼지제어기를 자동 생성하는 방법에 대하여 기술한다. 주어진 입출력 데이타를 표현하는 퍼지제어기는 각 유전자에 암호화되고, 퍼지제어기를 표현하는 각 유전자들은 서로 정보를 교환함으로써 주어진 데이타를 적절히 표현하는 퍼지제어기를 탐색하게 된다. 유전자는 각 입력변수의 언어항을 정의하고, 퍼지제어규칙은 정의된 언어항과 주어진 데이타로부터 생성된다. 탐색과정에서 퍼지제어기의 제어규칙과 각 입력변수의 언어항의 개수와 위치는 계속 변화하여 주어진 입출력 데이타를 잘 설명하는 퍼지제어기를 찾는다.

1 서론

퍼지제어는 퍼지이론의 응용분야 중 가장 활발히 연구되고 있다. 지금까지 퍼지제어기는 많은 분야에 적용되어 좋은 결과를 보이고 있으나 퍼지제어기의 개발과 미세조정에는 일반적인 방법론 보다는 대부분 전문가의 경험에 크게 의존하고 있다. 이러한 점을 보완하기 위하여, 퍼지제어기의 개발과 미세조정을 자동적으로 할 수 있는 많은 방법들이 연구되었으며, 최근에는 퍼지제어기와 유전자 알고리즘을 융합하는 연구들이 활발히 진행되고 있다.

유전자 알고리즘은 자연의 진화 현상을 모방해서 제안된 탐색 알고리즘이다. 유전자 알고리즘은 다른 탐색방법과 달리 대상이 되는 문제의 해를 문자열 형태로 표현하고, 그 문자열에 교차, 돌연변이 같은 자연의 생물유전을 모방한 연산자들을 반복 적용하여 적합한 해를 탐색한다. 유전자 알고리즘에서는 여러개의 개체가 공간상의 여러 점에서 서로 정보를 교환하며 최적해를 탐색하므로, 병렬처리가 가능하며, 또한 일정한 확률 범위 내에서 무작위 탐색이 진행 되므로 국소 최소값 문제를 어느 정도 완화할 수 있다는 장점을 갖고 있다.

[1] 한국과학기술원 전산학과 (Computer Science Department, KAIST)

이러한 유전자 알고리즘과 퍼지제어기의 융합에 관한 연구는 대부분이 전문가에 의해서 주어진 퍼지제어기나 퍼지분류기의 미세조정을 다루고 있다. 유전자 알고리즘에 의해서 자동 미세조정을 하는 경우, 전문가에 의해서 퍼지제어기가 주어지면, 그 제어기의 언어항의 소속함수나 제어규칙은 유전자 속에 암호화되고, 유전자 알고리즘에 의해서 언어항의 새로운 위치나 제어규칙을 찾아내게 된다. 그러나 이러한 방법은 이미 전문가에 의해서 개발된 퍼지제어기를, 진화연산자는 미세조정을 하므로 전반적으로 주어진 제어기를 크게 변형시키기 어렵다.

따라서, 어떤 시스템에 대한 정보가 주어졌을때, 전문가의 개입없이 그것으로부터 제어기를 생성하기 위해서는 미세조정과는 다른 방향의 연구가 필요하다. 본 논문에서는 어떤 시스템의 입출력 데이타가 주어졌을때, 이를 기반으로 유전자 알고리즘을 사용하여 퍼지제어기를 자동 생성하는 방법을 제시한다.

2 제안된 방법

주어진 데이타로부터 퍼지제어기를 자동생성하기 위해서는 각 유전자는 하나의 퍼지제어기를 표현해야 한다. 즉, 사전에 전문가에 의해서 퍼지제어기에 대한 정보가 주어지지 않으므로, 퍼지제어기의 정의에 필요한 모든 정보가 유전자에 암호화되어 있어야 한다.

하나의 퍼지제어기를 정의하기 위해서는 각 입력변수마다 언어항을 정의해야 하고, 정의된 언어항들을 이용하여 제어규칙을 생성해야한다. 따라서 하나의 유전자에는 각 입력변수에 정의된 언어항의 개수와 언어항의 정의, 정의된 언어항을 이용하여 생성된 제어규칙이 모두 포함되어야 한다. 그러나 이 모든 정보를 하나의 유전자에 암호화 한다면 유전자의 길이는 길어지게 되고, 따라서 탐색해야 될 공간이 확장되므로, 이를 해결할 수 있는 방법이 필요하다.

본 연구에서는 이러한 문제점을 해결하기 위하여 각 유전자에는 입력변수의 언어항 개수와 정의만을 암호화하고, 제어규칙은 정의된 언어항과 주어진 데이타로부터 자동생성하였다. 이 경우에 있어서 해결되어야 하는 몇가지 문제가 있다. 첫째는 입력변수마다 필요한 언어항의 개수가 전문가에 의해서 고정된 것이 아니라, 유전자 알고리즘에 의해서 각 입력변수마다 적절한 개수의 언어항이 찾아져야 하며, 둘째는 찾아진 언어항과 주어진 데이타로부터 제어규칙을 자동생성하는 방법이 있어야 한다. 이 절에서는 제안된 방법에서 사용한 제어규칙의 형태, 유전자 암호화규칙, 비퍼지화방법, 제어규칙의 생성, 목적함수에 관하여 기술한다.

제어규칙의 형태

퍼지제어기의 여러정보중 각 입력변수의 언어항에 관한 것만이 유전자에 저장되어 있고, 제어규칙은 주어진 데이타로부터 생성되어야 하므로, 제어규칙의 형태 역시 이를 반영하기 쉬워야 한다. 본 논문에서 사용한 제어규칙은 다음과 같이 조건부는 언어항으로, 결론부는 각 입력변수의 함수로 이루어진 것이다.

$$R_j : \text{If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then } output = F_j(x_1, \ldots, x_n)$$

$x_1, \ldots, x_n$은 입력변수이며, $A_{j1}, \ldots, Ajn$은 j번째 제어규칙의 언어항들이다.
결론부의 $F_j(x_1, \ldots, x_n)$은 $x_1, \ldots, x_n$의 이차함수이다.

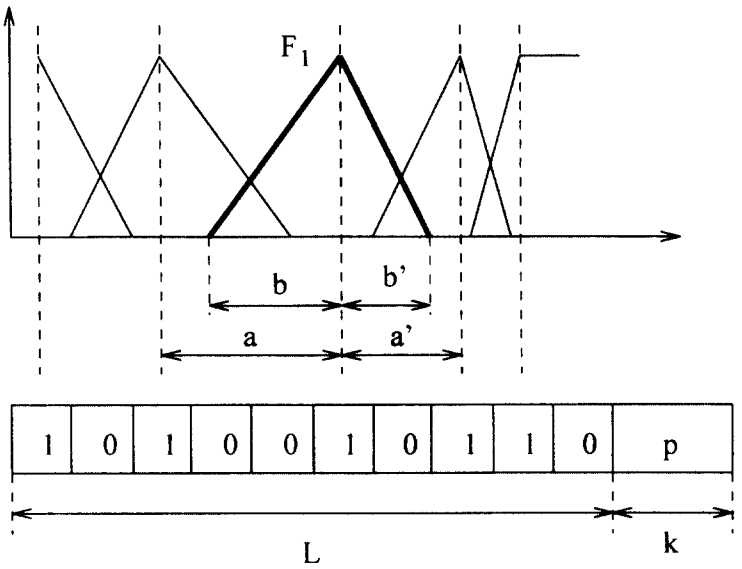

그림 1: 유전자의 암호화방식

암호화규칙

본 논문의 퍼지제어기는 계속되는 탐색동안 각 입력의 언어항과 위치가 계속 변하게 된다. 따라서 유전자 암호화방식은 이러한 것을 잘 반영할 수 있어야 한다. 기존의 유전자 암호화방식은 대부분 언어항의 개수가 정해져 있을때 그것들의 위치를 변화시키는데 적절하나, 언어항의 개수가 변화할 때에는 유전자의 길이가 변화하는 단점이 있다.

본 논문에서는 언어항의 소속도함수로 삼각 퍼지숫자를 사용하므로 한 언어항을 정의하기 위해서는 세개의 점이 필요하다. 그림 1과 같이 하나의 입력변수에 대하여 그것이 가질 수 있는 영역에 L개의 점을 설정하여 각 언어항의 꼭지점은 항상 그 점위에만 위치하도록 하였다. 즉, 유전자 속의 1에 해당 하는 위치에는 언어항이 존재하며, 0는 그 위치에는 언어항이 없음을 나타내게 된다. 또한 그 언어항의 양끝점의 길이는 언어항의 꼭지점에 이웃하는 언어항의 각 꼭지점까지의 거리의 p배가 된다. 그림 1에서 언어항 F_1의 좌측 끝점의 길이 b는 두 언어항 꼭지점 사이의 거리인 $a \times p$가 되는 것이다. 마찬가지로 오늘쪽 끝의 거리 b'역시 $a' \times p$이다. 따라서 하나의 입력변수마다 언어항의 위치를 나타내는 L개의 비트와 퍼지 삼각 숫자의 양끝을 결정하기 위한 p를 나타내는 k개의 비트가 필요하다. 입력변수가 n개이면 유전자 하나의 길이는 $n \times (L + k)$가 된다.

비퍼지화

한 입력이 주어졌을때 퍼지제어기의 출력을 결정하기 위해서는 입력으로부터 추론된 각 제어규칙의 결론부의 값을 결합하여 하나의 결론을 얻어내야 한다. 본 연구에서는 제어규칙의 결론부에 다항식을 사용하였으므로, 각 제어규칙의 결과값이 보통값(crisp value)이 되므로, 일반적인 비퍼지화 방법을 사용하지 않고, 다음과 같은 방식으로 최종 결과값을 얻었다.

입력이 $(x_1, \ldots, x_n)$이고, 제어규칙이 m개가 있을 경우, 입력에 대한 퍼지제어기의 출력은 아래와 같다.

$$output = \frac{\sum_{j=1}^{m} F_j(x_1, \ldots, x_n) \cdot \mu_j(x_1, \ldots, x_n)}{\sum \mu_j(x_1, \ldots, x_n)}$$

$$\mu_j(x_1, \ldots, x_n) = \min(\mu_{A_{j1}}(x_1), \ldots, \mu_{A_{jn}}(x_n))$$

μ_j는 주어진 입력이 j번째 규칙의 조건부를 만족시키는 정도이다.

제어규칙의 자동생성

본 논문에서 유전자가 하나의 퍼지제어기의 모든 정보를 갖고 있지 않고, 단순히 각 입력변수에 대한 언어항만을 정의하므로, 유전자 하나가 퍼지제어기 하나를 나타내기 위해서는 주어진 입출력 데이타와 유전자 속에 암호화된 언어항으로부터 제어규칙을 생성해 낼 수 있는 방법이 필요하다.

이러한 방법을 사용한 이유는 많은 정보를 유전자 속에 넣는 것은 유전자의 길이를 늘이게 되고, 자연적으로 탐색 공간을 넓히게 되어 전체적인 효율을 떨어뜨리게 된다. 또한 이미 입출력 데이타가 주어져 있으므로 이 데이타들을 단순히 제어기를 평가하는데 이용할뿐만아니라 제어기를 구성할 때도 이용할 수 있기 때문이다.

하나의 유전자로부터 제어규칙을 생성해 내는 방법은 다음과 같다. 먼저 각 입력의 언어항을 곱집합하여 가능한 모든 제어규칙의 조건부를 생성한다. 그리고 j번째 제어규칙의 결론부는 아래와 같이 구한다.

$$R_j : \text{If } x_1 \text{ is } A_{j1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{jn} \text{ then } output = F_j(x_1, \ldots, x_n)$$

주어진 입출력 데이타에서 다음과 같은 집합 S_j를 얻는다.

$$S_j = \{(x_{i1}, \ldots, x_{in}, y_i) | \mu_j(x_{i1}, \ldots, x_{in}) > 0, \quad 1 \le i \le n\}$$

n : 입출력 데이타의 수

얻어진 집합 S_j로부터 아래와 같은 $ESum$을 최소화 시키는 입력의 이차 함수 F_j를 얻는다.

$$ESum = \sum_{(X,y) \in S_j} (F_j(X) - y)^2, \quad X = (x_{i1}, \ldots, x_{in})$$

예를 들어 입력이 하나인 시스템에 그림 2와 같이 세개의 언어항과 입출력 데이타가 주어졌을 경우 j번째 제어규칙은

$$R_j : \text{If } x \text{ is } A_j \text{ then } output = ax^2 + bx + c$$

가 되며

$$S_j = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)\}$$

이다. 따라서,

$$ESum(a, b, c) = (ax_1^2 + bx_1 + c - y_1)^2 + (ax_2^2 + bx_2 + c - y_2)^2 +$$

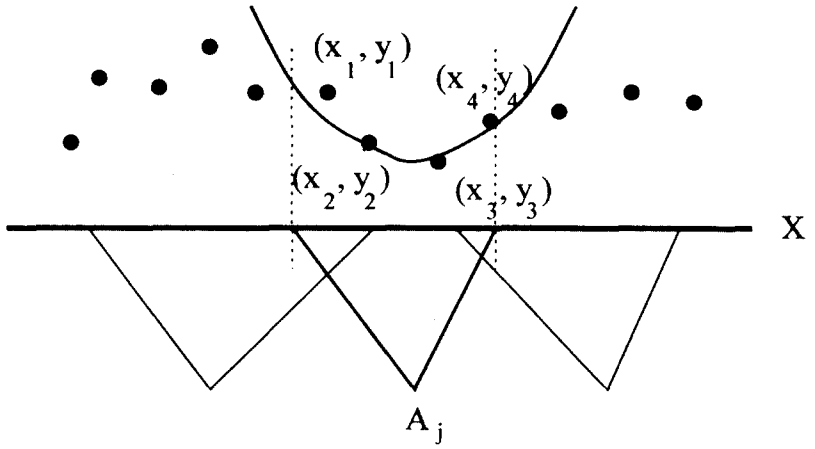

그림 2: 유전자와 퍼지집합의 관계

$$(ax_3^2 + bx_3 + c - y_3)^2 + (ax_4^2 + bx_4 + c - y_4)^2$$

이므로, 이것을 최소로 하는 a, b, c를 구한다.

목적함수

유전자 알고리즘에서 사용되는 목적함수는 한 유전자가 주어진 문제에 대하여 얼마나 좋은 해를 제공하는가를 판단하여야 한다. 본 연구에서는 한 유전자는 하나의 퍼지제어기를 나타내므로, 목적함수는 유전자에 의해서 표현된 제어기가 주어진 데이타를 얼마나 잘 근사하고 있는가를 판단해야한다. 따라서 제어기와 주어진 데이타와의 오차가 작을수록 좋은 해로 구별해야한다. 그러나 단순히 제어기의 오차가 작은 것만을 고려하게 되면 자연히 많은 제어규칙을 사용하는 제어기가 선택될 가능성이 높아지므로, 제어기가 사용하는 제어규칙의 수도 목적함수에 반영되어야 한다.

본 연구에서는 다음과 같은 목적함수를 정의하여 목적함수의 값이 작을수록 선택되어질 가능성이 커지도록 하였다.

$$cost = c_0 \times (ESum + n \times MErr) + c_1 \times m$$

$Esum$: 오차 제곱의 합
n : 주어진 데이타의 개수
$MErr$: 오차의 제곱 중 최대값
m : 제어규칙의 수
c_0, c_1 : 주어진 임의의 상수

3 실험 및 결과

실험은 입력이 하나인 시스템에 대하여 수행하였다. 각 언어항의 위치를 결정하는 L은 20으로 하고 각 언어항의 겹침정도를 결정하는 p에는 4비트를 할당하였다. 그리고 p의 값은 다음과 같

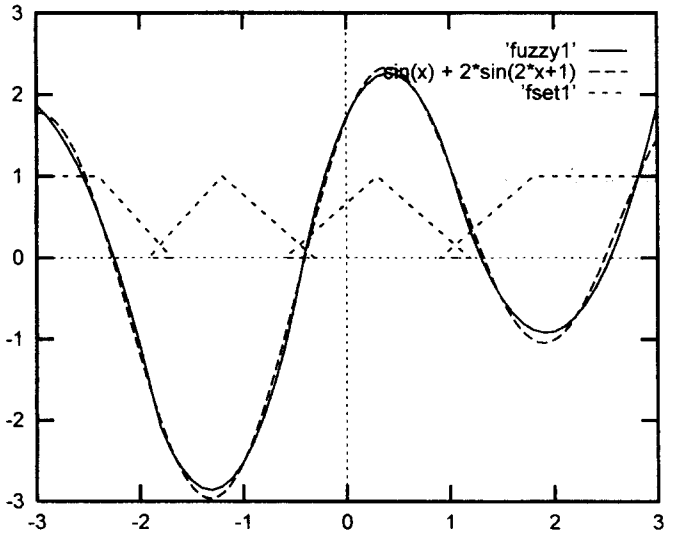

그림 3: $f_1(x)$의 근사

이 해석하였다.

$$p = 0.5 + 1.5(1 + 8p[0] + 4p[1] + 2p[2] + p[3])/16$$

위 식에 의하면 $p \geq 0.5$인데, 이것은 항상 한 언어항은 그 이웃에 있는 언어항과 겹치는 부분이 생기도록 하기 위함이다. 만약에 한 언어항이 이웃의 언어항과 겹치지 않는다면, 그 부분의 입력에 대해서는 해당되는 제어규칙이 생성되지 않기 때문이다. 교차확률은 0.6을, 돌연변이 확률은 0.05를 사용하였으며, 한 세대의 유전자 수는 10으로 하였다.

실험은 다음의 세 함수에 대하여 각각 수행하였다.

$$f_1(x) = \sin(x) + 2\sin(2x + 1)$$

$$f_2(x) = \begin{cases} 0.8(-2x + 1) + 0.1 & \text{if} x \leq 0.5, \\ 0.8(2x - 1)^2 + 0.1 & \text{otherwise.} \end{cases}$$

$$f_3(x) = (3(4x - 2)(4x - 3)(4x - 3.7)(4x - 1.3)(4x - 0.2) + 20)/40$$

그림 3, 4, 5는 각각의 결과 함수와 그 때의 입력의 언어항들을 나타낸다. 각 그림에서 실선은 생성된 퍼지제어기의 출력이며, 점선은 목적으로한 시스템이다. 각각의 경우에 목적함수의 c_0, c_1의 상수값은 $f_1(x)$, $f_2(x)$에서는 $c_0 = 1.0$, $c_1 = 0.5$이고, $f_3(x)$에서는 $c_0 = 3.0$, $c_1 = 0.1$이다. 각 그림에서 알 수 있듯이 제안된 방법은 전체적으로 적은 수의 제어규칙으로 목적함수를 대체적으로 잘 근사하였다. 그러나 $f_3(x)$의 경우에는 양쪽의 두 극점부분에서 오차가 심한 것을 발견할 수 있는데, 이것은 목적함수가 복잡한 모양을 가지고 있기 때문으로 판단된다. 이것을 해결하기 위해서는 입력변수의 영역을 더 미세하게 이산화해야 할 것이다.

4 결론

본 논문에서는 주어진 입출력 데이타로부터 진화적 방법을 이용하여 퍼지제어기를 자동 생성하

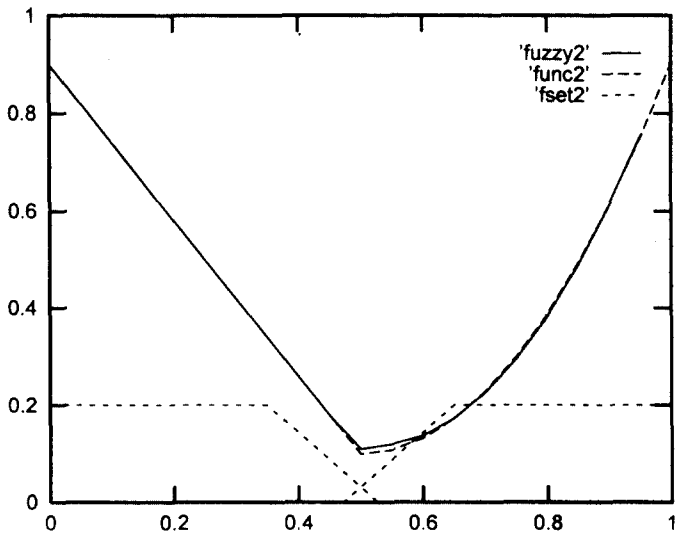

그림 4: $f_2(x)$의 근사

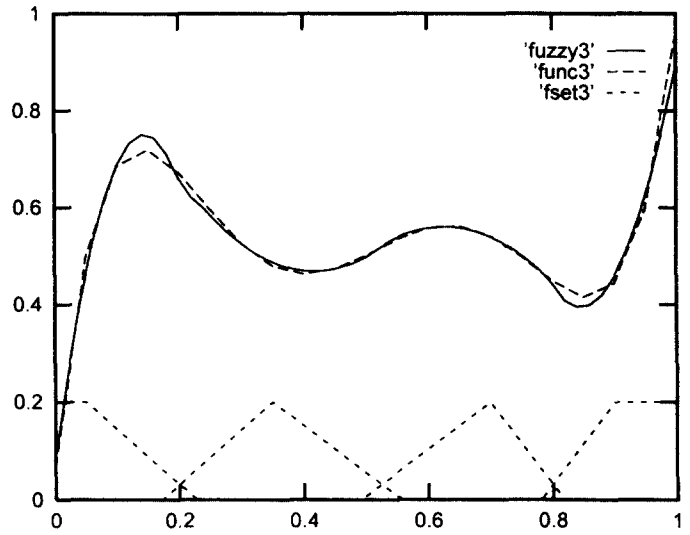

그림 5: $f_3(x)$의 근사

는 방법을 제안하였다. 제안한 방법은 각 유전자에 하나의 퍼지집합을 정의하여, 유전자사이에 진화연산자를 적용하므로써 주어진 데이타를 잘 표현하는 퍼지제어기를 생성했다. 입력이 하나인 시스템에 적용한 결과 전체적으로 작은 수의 제어규칙으로 주어진 입출력 데이타를 근사했으나, 모양이 복잡한 함수의 경우 오차가 커짐을 알 수 있었다. 이 문제에 대해서는 입력공간의 이산화 정도를 높이는 것과 유전자 알고리즘의 목적함수에서 오차에 대한 상수를 조절하는 방법이 있다.

따라서 앞으로, 이산화 정도와 시스템의 복잡도에대한 것과 일반적으로 적용될 수 있는 목적함수의 결정에 관한 것에 더 많은 연구가 필요하다.

참고 문헌

[1] C.L. Karr, E.J. Gentry, *Fuzzy Control of pH Using Genetic Algorithms*, IEEE Transactions on Fuzzy Systems, vol.7, No.1, pp.46-53, 1993.

[2] D. Park, A. Kandel, G. Langholz, *Genetic-Based New Fuzzy Reasoning Models with Application to Fuzzy Control*, IEEE Transactions on System, Man, and Cybernetics, vol.24, No.1, pp.39-47, 1994.

[3] P. Thrift, *Fuzzy Logic Synthesis with Genetic Algorithms*, Proceedings of the 4th International Conference on Genetic Algorithm, pp.509-513, 1991.

[4] F. Bolata, A. Nowé, *From Fuzzy Linguistic Specifications to Fuzzy Controllers using Evolution Strategies*, Proceedings of the 4th IEEE International Conference on Fuzzy Systems, pp.1089-1094, 1995.

[5] J. Kinzel, F. Klawonn, R. Kruse, *Modifications of Genetic Algorithms for Designing and Optimizing Fuzzy Controllers*, Proceedings of the 1st IEEE Conference on Evolutionary Computation, pp.28-33, 1994.

[6] T. Murata, H. Ishibuchi, *Adjusting Membership Functions of Fuzzy Classification Rule by Genetic Algorithms*, Proceedings of the 4th IEEE International Conference on Fuzzy Systems, pp.1819-1824, 1995.

[7] T. Furuhashi, K. Nakaoka, Y. Uchikawa, *An Efficients finding of Fuzzy Rules Using a New Approach to Genetic Based Machine Learning*, Proceedings of the 4th IEEE International Conference on Fuzzy Systems, pp.715-722, 1995.

[8] C-H. Chang, Y-C. Wu, *The Genetic Algorithm Based Tuning Method for Symmetric Membership Functions of Fuzzy Logic Control Systems*, 1995 International IEEE/IAS Conference on Industrial Automations and Control, pp.421-428, 1995.

[9] J. Kim, K-H. Lee, *A design and Implementation of the Fuzzy Logic Controller using the Adaptive Probabilities in Genetic Algorithm*, Proceedings of KFMS Spring Conference '95, vol.5, no.1, pp.202-208, 1995,(in korean).