문법 코딩을 이용한 그래프 구조 퍼지 시스템의 설계*

길 준 민°, 황 종 선
고려대학교 컴퓨터학과

A Design of Graph Structured Fuzzy Systems using Grammatic Coding

Joonmin Gil°, Chong-Sun Hwang
Dept. of Computer Science & Engineering, Korea Univ.

요 약

본 논문에서는 그래프 구조 퍼지 시스템을 유전자 알고리즘을 이용하여 최적화할 때, 해개체를 직접 코딩함으로써 발생되는 해개체 길이의 폭발적 증가 문제를 해결하기 위하여 문법 코딩 기법을 이용한 그래프 구조 퍼지 시스템을 제안한다. 문법적 코딩 기법은 퍼지 소속 함수와 퍼지 규칙의 상호 연관적인 규칙을 유전형으로 표현하여 퍼지 규칙의 반복적 패턴 혹은 재귀적 특성을 문법 규칙에 반영시킴으로써 유전자 알고리즘의 탐색 공간을 효율적으로 줄인다.

1. 서론

퍼지 시스템의 조직적인 설계를 위해서 퍼지 소속 함수의 조정, 퍼지 규칙의 개수 결정, 그리고 퍼지 규칙의 구조 결정 등의 고려 사항이 필요하다. 신경망의 학습 알고리즘 또는 유전자 알고리즘을 퍼지 시스템의 설계에 도입한 초기 방법론들은 주로 위의 세가지 고려 사항들을 부분적 혹은 상호 독립적으로 고려하였다[1-5]. 이는 퍼지 소속 함수와 퍼지 규칙의 연관성으로 인하여 부분적 혹은 상호 독립적으로 수행되는 퍼지 시스템의 설계 방법은 부분적인 최적화만을 고려한 설계 방법이라는 사고에 기초를 두고 있다. 또한 대부분의 제안된 방법들에서는 퍼지 규칙을 모든 퍼지 소속 함수의 조합 형태로 구성함으로써 고차원 입력 공간을 갖는 문제에 적용하였을 때 퍼지 규칙의 폭발적 증가 문제를 그대로 안고 있다.

이러한 문제점들의 인식 하에 [6-8]에서는 퍼지 규칙을 많은 탐색 공간을 야기시키는 규칙 테이블 형태로 표현하기보다는 입출력 관계를 적절히 표현하는 방법들을 제안하였다. 그러나 이 방법들은 입력 공간의 효율적인 감소를 위한 기틀을 제공함에도 불구하고 유전자 알고리즘이 퍼지 시스템을 최적화할 때 직접 코딩 기법을 이용하기 때문에 탐색 공간의 증가로 인한 계산 복잡도의 증가의 문제점을 가지고 있다. 이러한 문제점을 해결하기 위해서 본 논문에서는 퍼지 소속 함수와 퍼지 규칙의 구조를 생성시키는 문법 규칙을 유전자 알고리즘의 해개체로 코딩하는 문법적 코딩 기법에 기반하여 이를 최적화하고자 한다.

본 논문에서의 문법적 코딩 기법은 퍼지 소속 함수와 퍼지 규칙의 상호 연관적인 규칙을 유전형을 갖는 문법 규칙으로 표현하고 문법 규칙들의 연속적인 확장에 의해서 표현형을 갖는 퍼지 소속 함수와 퍼지 규칙의 구조를 생성한다. 또한 퍼지 규칙의 반복적 패턴 혹은 재귀적 특성을 문법 규칙에 반영시킴으로써 유전자 알고리즘의 탐색 공간을 효율적으로 줄인다.

본 논문의 구성은 다음과 같다. 2장에서는 선택적 퍼지 규칙에 기반한 그래프 구조 퍼지 시스템에 대해서 설명한다. 3장에서는 그래프 구조로 표현된 퍼지 시스템 내의 매개변수들을 문법적 코딩 기법을 이용하여 탐색 공간을 효율적으로 축소하는 방법에 대해서 기술한다. 4장에서는 본 논문에서 제안하는 방법의 유용성을 보이기 위한 실험 및 실험 결과를 보여준다. 마지막으로 5장에서는 결론을 기술한다.

2. 선택적 퍼지 규칙에 기반한 그래프 구조 퍼지 시스템

실제 퍼지 시스템을 설계할 때, 적용되는 문제에 따라서 모든 입력 공간에 대응되는 퍼지 규칙 대신에 입력 공간에 강한 반응을 보이는 퍼지 규칙만을 선택하는 경우가 있다. 퍼지 규칙의 구조를 규칙 테이블 형태로 표현하는 것이 이해하기 쉽지만 규칙 테이블로 표현된 퍼지 규칙의 구조는 모든 입력 공간을 완전히 포함한다. 따라서 입출력 관계를 최적으로 표현한 퍼지 규칙의 구조라고 볼 수 없다.

본 논문에서는 퍼지 규칙의 구조를 다음과 같이 정의하는 그래프 구조로 표현한다. 그래프 구조에서 노드는 하나의 루트 노드(root node)와 각 입력 변수에 대한 퍼지 소속 함수를 나타내는 언어항 노드(verbal node)들로 구성된다. 루트 노드와 각 언어항 노드 사이의 관계는 방향성을 갖는 에지로 표현된다. 또한 각 변수에 대한 언어항 노드는 하위 변수에 대한 언어항 노드로 방향성을 갖는 에지를 갖는다.

설명의 편의를 위해 입력 변수가 5개이고, 첫 번째 입력 변수에서 다섯 번째 입력 변수에 대한 퍼지 소속 함수의 개수가 각각 3개, 2개, 3개, 3개, 4개로 구성되는 경우를 생각해 본다. 그림 1은 이러한 경우에 대응하는 특정 그래프 구조를 보여준다.

그림 1에서 R은 루트 노드를 나타내며, A_j^i는 i번째 입력 변수에 대한 j번째 퍼지 소속 함수를 의미하는 언어항 노드를 나타낸다. 루트 노드에서 언어항 노드로의 에지는 점선으로 표현되며, 실선은 언어항

* 이 논문은 1997년 한국학술진흥재단 공모과제 연구비에 의해서 연구되었음

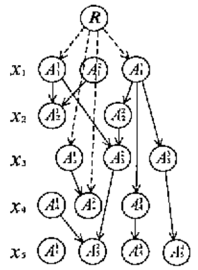


그림 1. 그래프 구조의 표현

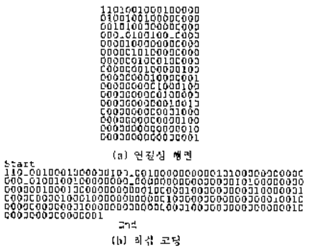


그림 2 직접 코딩 기법에 의한 그래프 구조의 해석

노드들 사이의 에지를 나타낸다.

그래프 구조에 대응되는 퍼지 규칙 구조로의 해석 방법은 루트 노드에서 최하위 입력 변수에 대한 언어항 노드들까지의 경로(path)를 찾는 것이다(자세한 설명은 [8]을 참조). 이와 같은 방식으로 생성되는 퍼지 규칙의 형태는 입력 공간 위에서 불필요한 입력 변수와 퍼지 소속 함수의 제거될 수 있는 형태를 띠고 있다. 즉, 퍼지 규칙을 각 입력 변수에 대한 모든 퍼지 소속 함수들의 조합으로 구성하는 형태가 아닌 선택적 조합의 형태로 나타냄으로써 주어진 문제에 최적의 퍼지 규칙을 생성할 수 있는 기틀을 제공한다.

3. 그래프 구조 퍼지 시스템을 위한 코딩 기법

3.1 그래프 구조의 코딩 방법

유전자 알고리즘을 이용하여 최적의 노드와 에지를 탐색하고자 할 때, 진화 탐색 시간과 탐색 공간의 크기는 주어진 문제의 적절한 코딩 방법에 의해서 좌우된다. 그림 2는 연결성 행렬(connectivity matrix)로 표현된 그래프 구조를 직접 코딩 기법으로 표현한 예이다. 연결성 행렬로부터 그래프 구조의 변환은 다음과 같다. 연결성 행렬의 인덱스가 $i=j$ 인 경우에는 노드의 존재에 대한 유무를 나타낸다. 이 경우 연결성 행렬의 값이 '1'이면 노드가 존재하는 것이고 '0'이면 노드가 존재하지 않는다. 연결성 행렬의 인덱스가 $i \neq j$ 인 경우에는 연결성 행렬에서 i번째 행과 j번째 열의 값의 해석에 의해서 그래프 구조에서 에지의 유무를 결정한다. 즉, 연결성 행렬의 값이 '1'이면 i 번째 노드에서 j번째 노드로 에지가 존재하는 것이고 '0'이면 에지가 존재하지 않는다 그림 2(b)의 해개체는 그림 2(a)의 연결성 행렬을 직접 코딩 기법에 의해서 표현한 형태이다 연결성 행렬을 직접 코딩 기법으로 코딩한다면 노드 수의 증가에 따라서 탐색 공간의 빠른 증가를 야기하고 유전자 알고리즘이 각 해개체를 평가하고 유전 연산자를 수행하는데 많은 계산 복잡도를 요구한다. 이러한 인식 하에 본 논문에서는 직접 코딩 기법 대신 그래프 구조를 생성시키는 문법 규칙을 코딩하는 접근 방법을 시도한다.

3.2 그래프 구조를 위한 문법 규칙

그래프 구조의 생성 방법은 스트링-재작성 메커니즘의 형태를 도입한 L-system에 기반한다[9]. 그래프 구조의 생성 과정은 시작 심볼(S)로부터 출발하여 2×2 행렬의 형태를 갖는 문법 규칙들의 연속적인 확장에 의해서 0과 1로 구성되는 연결성 행렬을 생성하고, 연결성 행렬의 해석에 의해서 그래프 구조를 구성한다. 그림 3은 그림 1의 그래프 구조를 생성시키는 문법 규칙들을 보여준다

문법 규칙은 다음과 같이 세 가지 종류로 분류된다 재귀적 문법 규칙(recursive grammatic rule), 비재귀적 문법 규칙(nonrecursive grammatic rule), 그리고 원시 문법 규칙(primitive grammatic rule) 재귀적 문법 규칙은 문법 규칙의 왼쪽 부분(LHS: left hand side)과 오른쪽 부분(RHS: right hand side)이 비종결 심볼의 2×2 행렬로 구성된다 이 문법 규칙은 재귀적 확장을 허용하는

$$S\rightarrow\begin{bmatrix} A & B \\ C & D \end{bmatrix} A\rightarrow\begin{bmatrix} E & F \\ H & I \end{bmatrix} B\rightarrow\begin{bmatrix} G & H \\ J & K \end{bmatrix} C\rightarrow\begin{bmatrix} H & H \\ H & H \end{bmatrix} D\rightarrow\begin{bmatrix} H & M \\ H & L \end{bmatrix}$$

(a) 재귀적 문법 규칙

$$E\rightarrow\begin{bmatrix} n & c \\ a & f \end{bmatrix} F\rightarrow\begin{bmatrix} d & f \\ f & a \end{bmatrix} G\rightarrow\begin{bmatrix} a & b \\ d & e \end{bmatrix} H\rightarrow\begin{bmatrix} a & a \\ a & a \end{bmatrix} I\rightarrow\begin{bmatrix} f & e \\ a & f \end{bmatrix}$$
$$J\rightarrow\begin{bmatrix} a & a \\ a & b \end{bmatrix} K\rightarrow\begin{bmatrix} a & a \\ e & a \end{bmatrix} L\rightarrow\begin{bmatrix} f & c \\ a & f \end{bmatrix} M\rightarrow\begin{bmatrix} e & c \\ a & d \end{bmatrix}$$

(b) 비재귀적 문법 규칙

$$a\rightarrow\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} b\rightarrow\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} c\rightarrow\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} d\rightarrow\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} e\rightarrow\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\cdot$$
$$n\rightarrow\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} o\rightarrow\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} p\rightarrow\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

(c) 원시 문법 규칙

그림 3. 그림 1의 그래프 구조를 위한 문법 규칙들

문법 규칙으로서 적은 수의 문법 규칙으로도 복잡한 구조의 그래프 구조를 쉽게 표현할 수 있다. 비재귀적 문법 규칙은 문법 규칙의 왼쪽 부분은 비종결 심볼로 구성되고 오른쪽 부분은 종결 심볼 중 원시 문법 규칙의 왼쪽 부분에 해당되는 심볼('a'~'p')의 2×2 행렬로 구성된다. 원시 문법 규칙은 그래프 구조 생성을 위한 마지막 확장에 필요한 문법 규칙으로서 문법 규칙의 왼쪽 부분은 종결 심볼로 구성되고 오른쪽 부분은 0과 1의 2×2 행렬로 구성된다.

그림 4는 그림 1의 그래프 구조에 대응되는 연결성 행렬을 얻기 위한 그림 3으로 표현된 문법 규칙의 확장 과정을 보여준다.

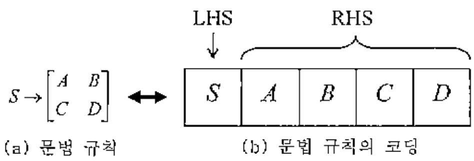
그림 4. 문법 규칙의 확장 과정

그래프 구조의 생성은 시작 심볼인 'S'로부터 시작한다(그림 4(a)) 첫 번째 확장 과정에서 초기 그래프는 그림 3의 첫 번째 규칙의 오른쪽 부분으로 대치되면서 재작성된다(그림 4(b)). 두 번째 확장 과정에서도 각 심볼들은 심볼에 대응되는 문법 규칙들의 오른쪽 부분으로 대치되면서 재작성되고 네 번째 확장 과정을 거치면서 최종 상태인 16×16의 연결성 행렬을 얻게 된다(그림 4(c)~(e)).

3.3 문법 규칙의 코딩 방법

그래프 구조에 대응되는 연결성 행렬을 얻기 위해서 본 논문에서는 그래프 구조 생성에 필요한 문법 규칙을 코딩하는 문법 코딩 기법을 유전자 알고리즘의 코딩 방법으로 도입하여 이를 최적화시킨다 문법 규칙 중 원시 문법 규칙은 16가지의 0과 1의 조합으로 구성되기 때문에 해개체로 코딩되지 않고, 문법 규칙과 비재귀적 문법 규칙만이 유전자 알고리즘의 해개체로 코딩된다.

문법 규칙을 위한 부해개체는 그림 5에서 보는 바와 같이 문법 규칙에서 왼쪽 부분의 한 심볼(LHS)과 오른쪽 부분의 네 심볼(RHS)들이 순차적으로 연결되어 구성된다.



(a) 문법 규칙 (b) 문법 규칙의 코딩

그림 5 문법 규칙을 위한 부해개체의 표현

연결성 행렬로부터 그래프 구조로의 변환은 그래프 구조를 생성시키는 문법 규칙을 나타내는 유전형과 그래프 구조를 나타내는 표현형의 변환이다. 유전형의 최적화는 특별한 형태의 그래프 구조, 예를 들어 모듈화된 구조 등을 다루는데 있어서 적은 문법 규칙으로도 충분히 표현되므로 표현형을 직접 최적화하는 것보다 효율적이다. 또한 그래프 구조에 대한 진화 과정의 이론적 분석을 위해서도 단순한 표현 방법을 갖는 유전형이 효율적일 것으로 기대된다.

3.4 직접 코딩과 문법 코딩 기법의 이론적 분석

퍼지 시스템 내의 매개 변수를 직접 코딩 기법과 문법 코딩 기법에 의해서 해개체로 표현할 때 요구되는 해개체의 길이를 살펴보자. 입력 공간이 d차원이고 각 입력 공간이 N개의 퍼지 소속 함수로 분할되었다고 가정한다.

직접 코딩 기법은 연결성 행렬을 직접 코딩하기 때문에 해개체의 길이는 $(d \cdot N)^2$이다. 반면, 문법 코딩 기법은 문법 규칙의 개수에 의해서 해개체의 길이가 좌우된다. 문법 규칙에 대한 확장의 개수가 1이면 1개의 문법 규칙이 필요하다 문법 규칙에 대한 확장의 개수가 2이면 문법 규칙의 개수는 첫 번째 확장을 위한 한 개의 문법 규칙과 두 번째 확장을 위한 4개의 문법 규칙이 필요하다 문법 규칙에 대한 확장의 개수가 3이면 첫 번째와 두 번째 확장을 위해서 5개의 문법 규칙이 필요하며 세 번째 확장을 위해서는 최악의 경우 16개의 문법 규칙이 필요하다(여기서 최악의 경우는 재귀적 패턴이 없는 경우로서 재귀적 패턴이 존재한다면 문법 규칙의 개수는 16개 미만이다). i번째 확장에 의한 문법 규칙의 개수는 i-1번째 확장에 기껏해야 16개의 문법 규칙만이 추가된다. 따라서 문법 규칙에 대한 확장의 개수를 w라 하면 문법 규칙의 개수는 $1+4+16(w-2)$이다. 하나의 문법 규칙은 5개의 유전인자가 필요하며 확장에 의한 퍼지 소속 함수의 개수는 $d \cdot N \leq 2^w$이므로 문법 규칙을 해개체로 코딩할 때의 해개체 길이는 $5 \cdot (16 \cdot \lceil \log_2(d \cdot N) \rceil - 17)$이다. 표 1은 해개체 길이에 대한 직접 코딩과 문법 코딩 기법의 복잡도를 보여준다

표 1. 직접 코딩과 문법 코딩 기법의 복잡도

	직접 코딩	문법 코딩
해개체의 길이	$O((d \cdot N))$	$O(\lceil \log_2(d \cdot N) \rceil)$

그래프 구조를 위한 문법 코딩 기법은 직접 코딩 기법보다 동일한 퍼지 규칙의 구조를 표현하면서 훨씬 적은 해개체의 길이가 필요하므로 탐색 공간을 효과적으로 축소할 수 있는 이점을 제공한다

4. 실험 및 실험 결과

본 논문에서 제안한 방법의 유용성을 보이기 위해서 제안한 방법을 시간열 예측 문제에 적용하였다 본 실험에서 시간열 예측 문제로서 사용하는 Mackey-Glass chaotic time series 문제는 다음과 같이 정의된 지연 미분 방정식에 의해서 생성된다:

$$\frac{dx(t)}{dt} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (1)$$

시간열 예측 문제는 과거의 값들 즉, $\{x(t), x(t-\Delta t), \cdots, x(t-n-1)\Delta t\}$로부터 미래의 값, 즉 $x(t+P)$를 예측하는 것이다 P에 대한 적당한 Δt 와 n의 선택에 따라서 입력 변수의 개수와 문게의 성질이 결정되는데, 본 실험에서는 $P=\Delta t=6$이고 $n=6$인 시간열 예측 문제를 다룬다. 각 점에서 시간열을 얻기 위한 방법으로, 4차 Runge-Kutta 방법을 이용하여 (1)의 수치적 해를 발견하였고, 시간 단계는 0.1, 초기 데이터 $x(0)$ 는 0.8, 그리고 τ 는 30을 사용하여 t가 130부터 1129까지 1000개의 데이터를 추출하였다 이 데이터 중 500개는 학습 데이터, 나머지 500개의 데이터는 시험 데이터로써 사용하였다. 각 변수에 대한 퍼지 소속 함수는 [12, 14]에서 정의되도록 하였으며, 문법 규칙을 위한 부해개체는 1개의 초기 문법 규칙과 각각 30개의 재귀적과 비재귀적 문법 규칙을 유전인자로 할당하였다.

표 2는 본 논문에서 제안한 문법 코딩 기법을 이용한 그래프 구조 퍼지 시스템과 직접 코딩 기법을 이용한 그래프 구조 퍼지 시스템에 의해서 생성된 결과를 비교한다.

표 2 Mackey-Glass chaotic 시간열 예측 문제에 대한 결과 비교

		직접 코딩 [8]	본 논문의 방법
퍼지 규칙		29	27
NMSE	학습 데이터	0.0509	0.0433
	시험 데이터	0.1637	0.1505

5. 결론

본 논문에서는 그래프 구조 퍼지 시스템을 유전자 알고리즘을 이용하여 최적화할 때, 해개체를 직접 코딩 기법에 의해서 표현함으로써 발생되는 해개체 길이의 폭발적 증가 문제를 해결하기 위하여 그래프 구조를 생성하는 문법 규칙을 해개체로 코딩하는 문법 코딩 기법을 이용한 그래프 구조 퍼지 시스템을 제안하였다 또한 제안된 방법을 시간열 예측 문제에 적용하여 직접 코딩에 의한 방법보다 더 적은 수의 퍼지 규칙만으로도 작은 NMSE를 생성함을 보여주었다.

참고문헌

[1] A. Homaifar and E McCormick, "Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms," IEEE Trans. on Fuzzy Systems, Vol. 3, No. 2, pp. 129-139, 1995.

[2] C. L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," Proc. of 4th Int. Conf. on Genetic Algorithms, pp 450-457, 1991

[3] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," Proc. of IEEE Int. Conf. on Fuzzy Systems, pp. 612-617, 1993.

[4] J Liska and S. S Melsheimer, "Complete design of fuzzy logic systems using genetic algorithms," Proc. of the 3rd IEEE Conf on Fuzzy Systems, pp. 1377-1382, 1994.

[5] H Nomura, I. Hayashi, and N. Wakami, "A self-tuning method of fuzzy reasoning by genetic algorithm," in Fuzzy Control Systems, A Kandel and G. Langholz, Editors, Baca Raton, FL.: CRC Press, pp 337-354, 1994

[6] K. Shimojima, T Fukuda, and Y. Hasegawa, "Self-tuning fuzzy modeling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm," Fuzzy Sets and Systems, vol 71, pp. 295-309, 1995.

[7] 김준민, 정창호, 강성훈, 박주영, 박대희, "수목구조 지능시스템을 이용한 고차원 공간 위에서의 비선형근사," 한국 퍼지 및 지능시스템 학회 논문지, 제 6권, 제 3호, pp. 25-36, 1996

[8] 김준민, 박대희, 박주영, "유전자 알고리즘을 이용한 고차원 입력 공간을 갖는 퍼지 시스템의 설계," 한국정보과학회 논문지, 제 24권, 4호, pp. 403-412, 1997

[9] P. Prusinkiewicz and A Lindenmayer, *The algorithmic beauty of plants*, Springer-Verlag, 1996