특 집

유전 알고리즘 이론 및 응용

張 炳 卓

建國大學校 컴퓨터工學科

I. 서 론

진화 알고리즘 (evolutionary algorithms) 또는 유전 알고리즘 (genetic algorithms)은 자연세계의 진화 현상에 기반한 계산 모델이다. 진화 알고리즘은 풀고자 하는 문제에 대한 가능한 해들을 정해진 형태의 자료 구조로 표현한 다음 이들을 점차적으로 변형함으로써 점점 더 좋은 해들을 생성한다. 각각의 가능한 해를 하나의 유기체 (organism) 또는 개체 (individual)로 보며 이들의 집합을 개체군 (population)이라 한다. 하나의 개체는 보통 한 개 또는 여러 개의 염색체 (chromosomes)로 구성되며 염색체를 변형하는 연산자들을 유전 연산자 (genetic operators)라 한다. 진화 알고리즘은 탐색, 최적화 및 기계학습의 도구로 많이 사용되었다. 구조가 단순하고 방법이 일반적이어서 유전 알고리즘의 응용 범위는 그 외에도 상당히 넓은 것이 특징이다.

염색체를 표현하는 방법과 사용되는 유전 연산자의 종류 및 특성에 따라 여러 가지 진화 알고리즘이 있는데 〈표 1〉에서는 대표적인 4가지 모델의 특징을 요약하고 있다. 협의의 유전 알고리즘 (genetic algorithm) 즉 GA는 고정된 길이의 이진스트링을 염색체로 사용한다 「Holland 1975, Goldberg 1989」. 이에 반해 진화 전략 (evolution strategy, ES)은 실수의 값을 취하는 유전자들로 구성된 벡터를 염색체로 사용한다 「Rechenberg 1973, Baeck & Schwefel 1993」. 그 밖에도 그래프와 트리를 염색체 표현에 사용하는 진화 프로그래밍 (evolutionary programming, EP)과 유전 프로그래밍 (genetic programming, GP) 등이 있다 「Fogel 1966, Fogel 1995, Koza 1992」. 진화적 탐색에 사용되는 연산자로 EP와 ES는 mutation, GA와 GP는 crossover를 주로 사용한다. 역사적으로 볼 때 EP, ES, GA는 1960년대와 70년대에 개발되었으며 GP는 90년대에 들어와 연구되기 시작한 분야이다.

진화 알고리즘에 관한 국제학회로는 ISGA (International Society for Genetic Algorithms)가

〈표 1〉 진화 알고리즘의 종류

이 름	문제 표현 방식			주연산자
진화 프로그래밍(EP)	이산치	그래프	크기 고정	mutation
진화 전략(ES)	실수	벡터	길이 고정	mutation
유전 알고리즘(GA)	0/1	스트링	길이 고정	crossover
유전 프로그래밍(GP)	기본함수	트리	크기 가변	crossover

있으며, 1985년부터 2년마다 홀수년에 ICGA(International Conference on Genetic Algorithms) 국제학술대회를 개최해 왔다. 전문 학술지로는 Evolutionary Computation 저널이 1993년부터 MIT Press에서 발간되고 있다. 1990년부터는 유럽을 중심으로 PPSN(Parallel Problem Solving from Nature) 학술대회가 매 짝수년에 열리고 있으며 1994년부터는 IEEE Computer Society 주최로 매년 ICEC(International Conference on Evolutionary Computing)가 개최되고 있다. 그 외에 EP 중심의 학술대회인 International Conference on Evolutionary Programming이 매년 열리고 있으며, 1996년에는 처음으로 GP 위주의 학회인 International Conference on Genetic Programming이 개최될 예정이다. 그 외에 IJCAI, AAAI, ECAI 등 인공지능 관련 학회와 학술지를 통해 진화 알고리즘에 대한 연구가 꾸준히 발표되어 왔다.

본고에서는 이들 진화 알고리즘 중 비교적 단순한 형태의 하나인 협의의 유전 알고리즘 즉 GA의 동작 원리와 그 이론적 기반에 초점을 두어 살펴보고자 한다. 다른 진화 알고리즘의 주요 연구 결과에 대해서는 응용사례와 참고문헌을 열거함으로써 대신하고자 한다.

II. 유전 알고리즘의 구조 및 동작 원리

모든 진화 알고리즘은 적어도 다음과 같은 5개의 구성 요소를 지닌다.

- 개체 표현 방법(encoding schemes)
- 유전 연산자(genetic operators)
- 적합 함수(fitness function)
- 선택 메카니즘(selection mechanism)
- 알고리즘 제어 파라메터(algorithm control parameters)

각각의 구성 요소에 대해서는 잠시 후 살펴보기로 하고 먼저 그 동작 원리를 알아보고자 한다.

전형적인 유전 알고리즘은 임의의 값으로 초기화된 개체들의 집합으로 시작한다(그림 1 참조). 각각의 개체는 상대적인 문제해결 능력에 따라 그 적합도(fitness)가 평가되며 적합도에 따라 다음 세대(generation)에 부모의 유전자가 복제(reproduce)되는 정도를 달리 함으로써 우성 형질을 지닌 개체들은 열성 형질을 지닌 개체들에 비하여 더욱 많은 자식을 생성할 수 있도록 유도된다. 이러한 선택(selection) 메카니즘은 다아윈의 진화론에서의 적자 생존의 원리(survival of the fittest)에 연유한다. 선택 복제된 개체들은 여러 가지 유전 연산자들에 의해 재결합(recombine)되어 다음 세대의 개체군을 형성한다. 이와 같은 세대 교체는 원하는 수준의 해가 개체군 내에 존재하거나 또는 다른 종료 조건이 만족될 때까지 반복된다.

```
procedure SimpleGeneticAlgorithm()
  initialize(Population);
  evaluate(Population);
  while not (termination condition satisfied) do
        MatingPool = reproduce(Population);
        Population = recombine(MatingPool);
        evaluate(Population);
  end while
end procedure
```

〈그림 1〉 기본적인 유전 알고리즘

유전 알고리즘이 기존의 탐색 또는 최적화 방법과 다른 점은 다음과 같다.

- GA는 점 (point)이 아닌 군 (population)에 기반한 탐색 방법이다.
- GA에서의 탐색은 확률적 연산자를 사용하여 수행된다.

– 탐색공간에 대해 연속성이나 미분가능성 등의 제약을 요구하지 않는다.

1. 개체 표현법

유전 알고리즘은 풀고자 하는 문제의 변수값을 이진 스트링으로 표현한다. 이 코딩 방법은 문제의 변수가 이진치이거나 다른 이산치를 갖는 경우에 특히 자연스러운 표현법이다. 만약 문제가 실수의 값을 가질 경우에는 이진화된 정수 형태로 변환하여 사용한다. 예를 들어, 어떤 변수가 −5.11에서 +5.12까지의 연속치를 가질 때, 이 값에 100을 곱하고 511를 더한 다음 이 정수값을 10비트로 표현함으로써, 소숫점 이하 둘째자리까지의 정확도를 가진 최적화를 시도할 수 있다.

좀 더 구체적으로, 문제의 한 변수에 대한 최소값 min과 최대값 max를 알고 이 변수의 값을 n개의 비트로 최적화하고자 할 때 다음과 같은 디코딩 방식에 의해 각 이진스트링에 대한 실수값을 구할 수 있다.

$$\mathrm{ival}_{\mathrm{bin}}\frac{(\max-\min)}{2^n-1}+\min$$

여기서 $\mathrm{ival}_{\mathrm{bin}}$은 이진스트링에 대한 십진수 정수의 값이고, n은 이 변수에 배정된 스트링의 길이이다. 예를 들어, 풀고자하는 문제가 두 개의 변수 x_1과 x_2를 포함한다고 하자. 변수 x_1이 5에서 20까지의 값을 갖고 이 변수에 대해 4개의 비트를(즉 x_1= 4) 배정하고, 변수 x_2는 −10에서 20.5까지의 값을 가지며 여기에 7개의 비트를(n_2=7) 배정하고자 한다. 1번부터 4번까지의 유전자는 x_1에 할당되고 5번부터 11번까지의 유전자는 변수 x_2에 할당되었다면, 이진스트링 01101011001에 대한 실수값은 다음과 같이 구할 수 있다.

$$\begin{array}{cc} 0\,1\,1\,0 & 1\,0\,1\,1\,0\,0\,1 \\ x_1 & x_2 \end{array}$$

$$x_1=0110 \rightarrow 6\frac{20-5}{2^4-1}+5=11$$

$$x_2=1011001 \rightarrow 89\frac{20.5+10}{2^7-1}-10=11.37$$

2. 유전 연산자

유전 알고리즘에서 사용되는 기본적인 연산자는 크로스오버(crossover)와 뮤테이션(mutation)이다. 크로스오버는 여러 가지 변형이 있으나 전형적인 경우에 개체군 중에서 두 개의 부모 스트링 i_1과 i_2가 무작위로 선택된다.

$$i_1=(i_{1,1}, i_{1,2},\ldots, i_{1,x}, i_{1,x+1},\ldots, i_{1,l-1}, i_{1,l})$$

$$i_2=(i_{2,1}, i_{2,2},\ldots, i_{2,x}, i_{2,x+1},\ldots, i_{2,l-1}, i_{2,l})$$

또한 1부터 $l-1$(l은 스트링의 길이) 까지의 유전자 위치 중 임의의 부분이 크로스오버 지점(crossover site) x로 선택된다. i_1과 i_2로부터 두 개의 자식 스트링 j_1과 j_2가 다음과 같이 생성된다.

$$j_1=(i_{1,1}, i_{1,2},\ldots, i_{1,x}, i_{2,x+1}, i_{2,x+2},\ldots, i_{2,l})$$

$$j_2=(i_{2,1}, i_{2,2},\ldots, i_{2,x}, i_{1,x+2},\ldots, i_{1,l})$$

여기서 j_1은 첫 번째 부모의 1부터 x번까지의 형질과 두 번째 부모의 $x+1$부터 l까지의 유전형질을 물려받았으며, j_2는 두 번째 부모의 1부터 x번까지의 형질과 첫 번째 부모의 $x+1$부터 l까지의 형질을 유전받았다. 예를 들어, i_1 = (11111111), i_2 = (00000000)인 경우 x=5가 선택되었다면 이로부터 생성되는 두 개의 개체는 j_1 = (11111000), j_2 = (00000111)가 된다. 크로스오버는 이 연산자가 적용되는 확률을 나타내는 파라메터인 p_c에 의하여 그 적용 빈도수가 제어된다.

뮤테이션 연산자는 각각의 비트에 대해 뮤테이션 확률 p_m에 따라 0은 1로 1은 0으로 변경하는 연산자이다. 스트링에 대한 뮤테이션은 스트링에 있는 각각의 비트에 대한 뮤테이션을 독립적으로 p_m의 확률로 적용함으로써 수행된다. 예를 들어, 스트링 j_1 = (11111000)을 뮤테이션한 결과두 번째 비트가 돌연변이를 일으킴으로써 j_1 = (10111000)가 될 수 있다. GA에서는 뮤테이션 연산자가 잃어버린 유전형질을 복구하기 위한 2차적인 연산자로 사용된다. 전형적인 p_m의 값은 0.05이하이다.

3. 적합 함수

목적함수(objective function) 즉 최적화(최대화 또는 최소화) 하고자 하는 함수는 각 개체의 적합도를 평가하는 기반이다. 그러나 목적함수의 값의 범위는 문제마다 다르기 때문에 보통 정해진 구간 사이의 양수의 값을 갖도록 표준화된 값을 적합도로 사용한다. 엄격히 구별하자면, 표준화하기 전의 적합도의 값을 raw fitness라고 하며 표준화되어서 실제로 개체 선택의 기준이 되는 함수를 적합함수라고 한다.

표준화하는 한 가지 방법은 목적함수의 값을 다음과 같이 선형적으로 재조정되는 것이다.

$$f' = af + b$$

여기서 f는 raw fitness를 나타내고 f'는 표준화된 적합함수의 값을 나타낸다. 상수 a와 b는 알고리즘 수행의 처음부터 끝까지 고정된 값을 가질 수도 있고 매 세대마다 재조정될 수도 있다.

표준화하는 또 다른 방법으로서는 계산된 적합도의 표준편차를 고려하는 것이다. 개체군의 평균 적합도의 값을 $\bar{f}$, 표준편차를 q라 할 때 다음의 식에 의해 스케일링된다.

$$f' = f - (\bar{f} - cq)$$

상수 c는 보통 1부터 3 사이의 한 값을 취한다.

4. 선택 메카니즘

선택 연산자는 잘 적응한 해들은 살아남고 잘 적응하지 못한 해들은 도태되로록 유도함으로써 자연선택(natural selection) 현상을 모델링한다. 보통 이진 스트링을 사용하는 유전 알고리즘에서는 적합도 값에 비례하여 다음 세대에의 복제가 진행되는데 이를 적합도 비례 선택(fitness proportionate selection) 이라 한다. 세대 g에 존재하는 개체들의 평균 적합도의 값이 $\bar{f}(g)$라 할 때, 적합도 $f(i,g)$를 가진 부모 개체 i는 평균적으로 $\frac{f(i,g)}{\bar{f}(g)}$ 개의 자식을 생성한다. 따라서 평균치 이상의 적합도를 가진 개체는 한 개 이상의 자식을 복제할 수 있으며, 평균치 이하의 적합도를 가진 개체는 자식을 복제하지 못할 확률이 높다. 그러나 아무리 적합도가 낮은 개체일지라도 $\frac{f(i,g)}{\bar{f}(g)}$의 값이 항상 0보다는 크기 때문에 자식을 복제할 가능성이 전혀 없는 것은 아니다. 이러한 선택법은 개체군의 다양성이 급속도로 저하되는 것을 방지함으로써 좋은 해를 찾기 전에 수렴(premature convergence) 하는 것을 피할 수 있다.

5. 알고리즘 제어 파라메터

GA에 의한 탐색에 있어서는 기존의 탐색 결과를 최대한 이용하려는 경향 즉 exploitation과, 새로운 영역을 탐색하고자 하는 성질 즉 exploration의 적절한 결합이 중요하다. Exploitation이 상대적으로 강조될수록 기존의 hill-climbing 탐색법과 유사해지며, exploration이 강조될수록 random search와 같은 특성을 나타내게 된다. 이를 제어하기 위한 파라메터에는 여러 가지가 있으나 중요한 것 3가지만을 들자면 개체군의 크기 M, 크로스오버 확률 p_c 그리고 뮤테이션 확률 p_m이다.

일반적으로 큰 p_c와 p_m의 값은 알고리즘의 exploration 능력을 향상시킴으로써 진화의 초기에 적합도가 높은 탐색공간을 찾는데 유리하게 작용하지만, 동시에 exploitation 능력을 저하시킴으로써 어느 정도 좋은 해를 찾은 후에는 이 탐색 공간 내에서 최적해로 수렴하는데 있어서는 수렴 속도를 저하시키는 요인으로 작용할 수 있다. 작은 p_c와 p_m의 값은 반대의 특성을 나타낸다. 개체군의 크기 M이 작으면 적합도 계산에 필요한 시간을 절약할 수 있으나 개체간의 다양성의 빠른 손실로 인해 최적의 해를 구하기 전에 수렴할 위험성을 내포한다. 반면에 개체군의 크기가 크면 최적해에 도달할 확률은 높으나 많은 기억 용량과 계산시간을 필요로 한다. 이 둘의 성능 평가 항목을 모두 만족시켜 주는 최적의 개체군 크기를 정하는 방법은 문제의 성격과 다른 제어 파라메터들의 값에 따라 다르다.

III. 이론적 기반

실제 응용에 있어서의 많은 성공에도 불구하고 진화 알고리즘의 이론적인 연구에 대한 발전은 그리 빠르다고 볼 수 없는데, 여기에서는 GA에서의 이론적 기반이 되는 스키마 정리에 대하여 고찰해 보기로 한다. 이를 위해 먼저 스키마와 빌딩블록에 대한 개념을 알아보기로 한다.

1. 스키마와 빌딩블록

스키마(schema)는 정해진 스트링 위치에 같은 비트값을 가진 모든 가능한 스트링들의 부분집합이다. 스키마는 기호 0 또는 1의 어떤 값도 취할 수 don't care 기호 * 를 사용하여 표시한다. 가령 4개의 비트를 가진 스트링에서 스키마 $s=1**0$은 첫번째 유전자 위치에 1의 값을 그리고 4번째 유전자의 위치에 0의 값을 가지는 모든 스트링들을 나타낸다.

$$s=1**0=\{1000, 1010, 1100, 1110\}$$

스키마에 의해 표현되는 각각의 스트링들은 스키마의 인스턴스라 한다. 스키마에서 0 또는 1의 값을 가진 유전자의 위치를 고정위치(fixed position)라 한다. 고정위치의 갯수는 그 스키마의 오더(order)로 정의한다. 그리고 스트링의 가장 왼쪽 고정위치와 가장 오른쪽 고정위치 사이의 거리를 스키마의 길이(defining length)라 한다. 위 예의 스키마 $s=1**0$은 오더가 2이고 길이가 3이다.

스키마도 각각의 스트링처럼 적합도를 가지며, 스키마의 적합도는 이를 구성하는 각각의 요소 스트링들의 적합도의 평균값이다. 즉 g세대에 있어서 스키마 s의 적합도 $f(s, g)$는 다음과 같이 정의된다.

$$f(s,g)=\frac{\sum_{i\in I_s} f(i,g)}{N(s, g)}$$

여기서 I_s는 스키마 s에 속하는 개체들의 인덱스 집합이고 $f(i,g)$는 개체 i의 적합도이다. $N(s,g)$는 스키마 s의 인스턴스 개수이다.

GA 이론에 있어서 또하나 중요한 개념은 소위 빌딩블록(building blocks)이다. 빌딩블록은 스키마 중에서 적합도가 높고 스키마의 길이가 짧은 것을 가리킨다. 유전 알고리즘에 의한 최적의 스트링을 탐색하는 과정은 여러 스키마들간의 경쟁 과정으로 볼 수 있으며 결국 최적의 스트링을 인스턴스로 가진 스키마 s가 다른 스키마들에 비해 더욱 많은 자식을 복제하고 선택과정에서 살아남음으로써 개체군에서 s의 인스턴스의 개수가 늘어나는 과정으로 볼 수 있다. 그런데 만약 최적의 스트링이 고적합도의 짧은 스키마들 즉 빌딩블록들을 일렬로 나열하여 얻을 수 있다면, 개개의 빌딩블록들이 경쟁관계에서 이겨내기만 한다면 이들의 결합으로 최적의 스트링을 구축할 수 있다. 이와 같이 높은 적합도를 가진 스트링들이 높은 적합도를 가진 빌딩블록들을 찾아내어 이들을 결합함으로써 만들어질 수 있다는 전제를 빌딩블록가설(building blocks hypothesis) 이라고 한다.

2. 스키마 정리

유전 연산자의 적용에 의해 스키마의 인스턴스의 개수는 변한다. 먼저 복제(reproduction) 연산자에 의한 스키마 인스턴스 개수의 변화를 살펴본 후 크로스오버와 뮤테이션의 영향을 분석한다.

분석의 편의상 복제되는 자식의 개수가 부모 개체의 적합도에 비례한다고 가정한다. 즉 i번째 스트링에 의해 정확히 $f(i, g)/\bar{f}(g)$개 만큼의 스트링이 복제된다고 하자. 세대 g에 있어서의 스키마 s의 인스턴스의 개수를 $N(s,g)$로 표기하면 다음세대 $g+1$에서의 s의 인스턴스의 개수는 다음과 같다.

$$N(s, g+1)=\sum_{i\in I_s}\frac{f(i, g)}{\bar{f}(g)}=\frac{\sum_{i\in I_s} f(i,g)}{\bar{f}(g)}$$

여기서 I_s는 스키마 s의 인스턴스에 대한 인덱스 집합이다. $f(i, g)$는 인스턴스 i의 세대 g에서의 적합도이고, $\bar{f}(g)$는 g세대 개체군의 평균 적합도

이다. 또한 $f(s,g) = f(s,g) = \frac{\sum_{i \in I_s} f(i,g)}{N(s,g)}$ 로부터 다음의 식이 성립된다.

$$\sum_{i \in I_s} f(i,g) = N(s,g) f(s,g)$$

따라서 선택만에 의한 스키마의 인스턴스의 개수는 세대교체를 함에 따라 다음과 같이 변경된다.

$$N(s,g+1) = \frac{\sum_{i \in I_s} f(i,g)}{f(g)} = \frac{f(s,g)}{f(g)} N(s,g)$$

이 식에서 만약 $\frac{f(s,g)}{f(s)} = k$ 이면 $N(s,g) = k^g N(s,0)$가 되므로, 개체군 평균 적합도보다 높은 적합도의 값을 가진 스키마의 개수가 지수함수적으로 증가한다는 것을 알 수 있다.

다음에는 크로스오버와 뮤테이션이 스키마의 인스턴스의 갯수에 미치는 영향을 생각하기로 한다. 위에서 살펴본 바와 같이 크로스오버 연산자는 스트링의 임의의 한 부분을 분할하여 다른 스트링과 교환한다. 따라서 스키마의 길이 $\delta(s)$가 길면, 크로스오버의 위치가 스키마의 고정위치 사이로 선택되어 인스턴스가 파괴될 가능성이 높아진다. 즉 l이 스트링에 있는 비트의 개수를 나타낸다면, 스키마 s의 인스턴스가 p_c의 확률로 적용되는 크로스오버에 의해 파괴될 확률은

$$p_c \frac{\delta(s)}{l-1}$$

에 비례한다. 비슷하게, 한 스키마의 인스턴스의 개수는 스키마의 오더 $o(s)$가 클수록 뮤테이션에 의하여 파괴될 확률이 높다. 즉 스키마의 인스턴스의 개수는 뮤테이션 확률 p_m과 스키마의 오더 $o(s)$의 곱

$$p_m o(s)$$

에 비례하여 감소한다.

위의 결과를 종합하여 선택, 크로스오버 및 뮤테이션 연산자에 의한 영향을 모두 감안하면, 다음세대 $g+1$에서의 s의 인스턴스의 개수는 다음과 같이 변경된다.

$$N(s,g+1) \geq N(s,g) \frac{f(s,g)}{f(s)} \left[1 - p_c \frac{\delta(s)}{l-1} - p_m o(s) \right]$$

이 관계식은 GA에 대한 기본 정리를 제공하는 데 흔히 스키마 정리라 불리운다. 이 정리가 의미하는 바는 짧고 오더가 낮으며 평균치 이상의 적합도를 가진 스키마의 개수는 지수함수적으로 증가한다는 것이다.

스키마 정리는 몇가지 제한적인 상황하에서 유도된 것임에 주의하여야 한다. 가령 $f(s,g)/f(g)$가 $f(s)/f(g)$에 대한 정확한 추정치라고 가정하지만 실제 상황에서는 개체군의 크기가 제한되어 있고 샘플링시의 바이어스로 인하여 오차가 발생한다. 또한 크로스오버가 기존의 스키마를 파괴적인 연산자로 취급되나 이는 크로스오버가 탐색하는데 있어서 실제로는 유용한 연산자라는 실험적인 결과와 배치된다. 이와 같은 몇가지 제한점에도 불구하고 스키마 정리는 유전 알고리즘의 동작에 대한 하나의 이론적인 근거를 제공한다는 점에서 큰 의의를 지닌다.

IV. 응용 사례 : 신경망의 진화적 설계 및 학습

지금까지 살펴본 협의의 유전 알고리즘 즉 GA를 포함한 여러 진화 알고리즘에 의한 탐색은 탐색공간에 대한 제약, 예를 들어 공간의 연속성과 같은 제약조건을 갖지 않기 때문에 여러가지 문제에 적용이 가능하고 또 주어진 상황에 적응적으로 대처해서 탐색을 한다는 특징을 가지고 있다. 특히 multimodal 공간에서의 탐색, 최적화 및 학습 문제에 탁월한 성능을 보일 수 있음이 여러 연구에 의하여 확인되었다. 이러한 특성으로 인하여 진화 알고리즘은 여러 산업 및 경제 분야에 적용되고 있다. 몇 가지 대표적인 응용의 예가 〈표 2〉에 열거되어 있다.

〈표 2〉 진화 알고리즘의 응용 분야

응용 분야	응 용 사 례
최적화	수치적 함수의 최적화, 가스 파이프라인의 최적화, 전력 송전망의 최적화, 컴퓨터 자판의 최적 배정 문제, 항공기 승무원 배정 문제
설계	VLSI 회로 설계, 비행기 날개의 공기역학적 설계, 엔진 노즐의 설계, 컴퓨터 통신망의 최적 설계, 심장박동기의 설계
인공지능	LISP 프로그램의 자동 생성, 문제해결 규칙의 자동 습득, 신경망 합성 및 학습, 패턴 인식, 자연언어 처리, 멀티에이전트 시스템
시스템 분석 및 예측	시스템 동정, 케이오틱 시계열의 예측, 환률 변화 예측, 단백질 구조 분석, 재정 및 경제 분야에의 예측 및 분석 문제
제어 및 로보틱스	broom balancing, truck backer upper problem, 이동 로봇의 경 로 계획, 신경망 및 퍼지로직과 유전알고리즘의 결합에 의한 제어

특히 최근 들어서 퍼지시스템이나 신경망과 결합하여 응용되는 예가 늘어나고 있다. 퍼지시스템과의 결합은 세 가지로 대별할 수 있는데

1. 퍼지 멤버쉽 함수의 최적화
2. 퍼지 규칙 집합의 학습
3. 퍼지 멤버쉽 함수와 퍼지 규칙의 동시 학습

등이 있다「Carse et al. 1995」. 진화 알고리즘을 신경망 연구에 사용하는 방법도 여러가지가 제안되었는데 이는 다시 세부류로 나누어 볼 수 있다「Schaffer et al. 1992, Zhang 1992」.

1. 신경망 학습에 필요한 데이터의 최적화
2. 신경망의 연결강도를 학습하는 알고리즘으로 사용
3. 신경망 구조의 설계에 이용

첫번째의 경우는 신경망 학습의 전처리로서 유전 알고리즘을 사용하는 것이다. 가령 학습 데이터의 입력 변수의 갯수를 줄이거나 유효한 트레이닝 집합을 선택하는데 진화계산 개념이 적용된 바 있다「Zhang 1994」. 두 번째의 방법은 기존의 신경망 학습 알고리즘 대신 유전알고리즘을 사용하는 것인데 대개는 기존의 방법과 결합하여 사용된다. 특히 오차에 대한 미분값이 존재하지 않을 경우에 진화알고리즘은 아주 유용한 학습 알고리즘이다. 신경망 구조의 최적화를 위해서는 많은 방법들이 제안되었는데 이들은 대부분 스트링이나 매트릭스와 같은 형태로 망의 구조를 코딩하고 이에 유전연산자를 적용함으로써 최적화를 시도한다.

이에 반해「Zhang & Muehlenbein 1994」에서 제안된 방법에서는 유전 프로그래밍(GP)에서 착안하여 신경트리라 명명한 트리구조의 신경망을 사용하는데 그 크기가 동적으로 변하는 점이 특징이다. 또한 신경망에 응용된 기존의 진화 알고리즘은 대부분이 망의 구조나 연결강도의 최적화 중 하나를 목표로한 것에 반해, 신경트리에 의한 방법은 망의 구조와 웨이트의 값 뿐만 아니라 각 유닛의 종류까지 문제에 적합하게 적응시킬 수 있게 확장되었으며, 특히 이들이 상호 유기적으로 그리고 동적으로 유전연산자에 의해 최적화되도록 설계되었다.

신경트리는 인공신경들이 가변적인 연결선을 통해 트리형태로 결합된 구조이다. 트리의 잎은 외부 입력 x_i를 나타내며, n개의 변수로 구성된 터미널 집합 X의 원소이다. 트리의 루트노드는 신경트리의 출력을 나타낸다. 각 내부노드는 유닛타입 u_i, 전이함수f_i, 리셉티브필드 $R(i)$, 그리고 웨이트벡터 w_i를 가진다. $R(i)$는 i번째 유닛에 입력을 제공하는 유닛들의 인덱스집합을 나타낸다. 신경트리는 하나의 망구조에 서로 다른 종류의 신경유닛을 포함할 수 있다. 가능한 신경유닛은 응용분야에 따라 정의될 수 있으며 구체적인 실현은 진화적 학습에 의해 결정된다. 예를들어, 시그마유닛과 파이유닛을 함께 사용함으로써 여러 개의 은닉층을 사용하지 않고도 복잡한 함수구조를 효과적으로 학습할

수 있다.

주어진 문제에 적합한 신경망 모델을 구축하기 위해 M개의 신경트리로 구성된 개체군 A를 유지한다. 처음에는 트리가 임의의 크기와 모양으로 초기화된다. 각각의 신경트리는 아래에 기술되는 평가함수에 의해 적합도가 계산되며 상위의 %가 메이팅풀 에 선택된다. 다음 세대의 M개의 개체는 기존의 신경트리의 일부를 다른 신경트리와 치환함으로써(crossover) 신경트리의 구조와 크기를 변화시킨다. 신경유닛의 유형과 입력유닛의 인덱스는 돌연변이(mutation) 연산자에 의해서 변형된다. 선택때 가장 좋은 성능을 가진 개체는 항상 다음 세대에 살아남도록 보장하는 엘리트선택법(elitist strategy)을 사용한다. 그 이유는 세대교체의 궁극적인 목적이 가장좋은 성능을 가진 개체를 찾아내는데 있는데 반해 각각의 연산자는 국부적인 성능의 저하를 가져올 수 있기 때문에 이로 인해 세대교체가 일어남에 따라 그 세대까지의 최적의 해가 소실되는 것을 방지하기 위해서이다.

새로운 구조의 신경트리가 유전 연산자에 의해 생성되면 각각의 트리는 웨이트의 값들이 조정된다. 이 국부적 학습은 BGA「Muehlenbein 1993」에 의해 다음과 같이 웨이트의 값을 변경한다.

$$\Delta w = R \cdot 2^{-\varepsilon \cdot K}$$

여기서 K는 상수이며 ε은 0과 1 사이의 임의의 값이다. 이 방법은 다양한 범위의 최적화 문제에 있어서 효과가 있음이 입증되었다. 각 신경트리의 적합도는(최소화 문제의 경우) 다음과 같은 비용함수로 평가된다.

$$F_i = F(D/A_i) = \frac{E(D/A_i)}{m \cdot N} + \frac{C(A_i)}{N \cdot C_{max}}$$

위 식에서 m은 출력 유닛의 갯수이고 N은 학습에 사용된 예의 갯수이다. 첫번째 항은 학습집합 D에 대한 오차 E를 반영하고 두번째 항은 트리의 복잡도 C를 반영한다. 이 평가함수는 단순한 트리를 복잡한 트리에 대해 더 높이 평가함(적은 F_i 값)으로써 같은 오차를 가진 신경망의 경우 단순한 구조를 가진 신경망이 우선권을 가지고 선택될 가능성을 높여준다. 이렇게 함으로써 학습 데이터에 대한 오차를 줄이기 위해 트리의 크기가 점점 커지려는 경향을 줄여 주어 학습된 모델의 일반화 능력을 향상시킬 수 있다「Zhang & Muehlenbein 1995」.

신경트리의 진화에 의한 신경망 구조의 설계 및 학습법은 여러가지 예측 및 진단 시스템의 자동 설계 및 학습에 사용되었다「Zhang & Muehlenbein 1996」. 한 가지 응용 예는 환경 오염도의 예측 문제인데 학습 데이터는 일리노이의 상가몬강에서 1970년 1월 1일부터 1971년 12월 31일까지의 기간 동안에 강물에 포함된 질산의 함량 변화에 대한 시계열 자료이다. 학습의 목적은 지난 몇주 동안의 함량으로부터 그 다음 주의 질산의 양을 예측하는 것이다. 예측된 농도는 수질의 오염도를 모니터링하고 제어하는데 사용된다. 트레이닝 데이터는 함량 변화에 대한 시계열 자료로부터 최근 4주간의 변화를 입력해서 다음주의 값을 예상하는 것으로 생각하여 생성되었다. 이 예측 문제를 풀기 위해 초기의 신경트리들은 각 유닛의 최대 입력의 갯수가 4개, 그리고 트리의 최대 깊이가 4로 무작위로 생성되었다. 신경유닛의 50%는 시그마뉴런 나머지 50%는 파이뉴런으로 초기화되었다. 각 웨이트의 값은 [−10, +10]의 범위 내에서 조정되었으며, 300번의 세대 교체 후의 가장 좋은 결과는 3개의 은닉층에 10개의 은닉 신경유닛을 가진 신경망에 의해 얻을 수 있었다. 트레이닝 집합에 대한 학습 결과 그 평균자승오차는 0.0104였다. 진화에 의해 학습된 신경트리모델의 실제적인 예측능력을 검토하기 위해 학습 때 알려주지 않은 1972년의 자료를 사용하여 그 성능을 평가하였다. 그 결과 전통적인 방법인 GMDH에 의해 얻은 예측치보다 보다 더 정확한 값을 얻을 수 있었다.

V. 요약 및 전망

본고에서는 유전 알고리즘의 구성과 동작 원리 및 이론적 기반에 대해 살펴 보았다. 그리고 하나

의 응용 예로서 진화 알고리즘을 써서 신경망의 구조를 최적화하고 학습시키는 방법에 대하여 기술하였다. 이글을 맺기 전에 한 가지 언급되어야 할 사항은, 여기에 기술된 내용이 유전 알고리즘 또는 진화 알고리즘의 전부는 아니라는 것이다. 위에 살펴본 것은 단지 여러 가지 진화 계산 모델 중에 비교적 단순한 형태 중의 하나인 GA에서의 기본적인 사항들이며, 실제 응용되는 데에 있어서는 다른 여러 모델들이 개발되어 있으며 또한 계속 개발 변형되고 있다. 예를 들자면개체 선택에 있어서 위에서 기술한 적합도값에 비례한 선택법 외에, 상위 일정 비율의 개체를 선택하여 그들간에 무작위로 재결합하는 트렁케이션 선택법(truncation selection) 이나 또는 개체군 중에 k개를 임의로 선발하여 이중 가장 좋은 개체를 하나 선택하는 토너먼트 선택법(tournament selection) 등이 있다.

진화 알고리즘은 다양한 분야에 적용되고 있는데 이와 같은 많은 응용의 배경에는 몇 가지 이유가 있다. 그 한 가지는 알고리즘의 단순성과 일반성에서 찾을 수 있다. 아무리 복잡한 문제라도 일단 염색체 형태로 표현이 되면, 염색체의 복제와 재결합 및 적합도의 평가 등과 같은 비교적 단순한 연산 과정의 반복을 통해 계산이 수행된다. 또 하나의 이유는 기존의 문제 해결 방법과 결합하여 사용하기가 쉽다는 것이다. 이것은 진화 알고리즘이 문제 해결에 특수한 정보를 많이 사용하지 않고 또한 문제에 대한 배경지식이 있으면 이를 쉽게 수용할 수 있기 때문이다. 예를 들어 문제 P에 대한 기존의 경험적 알고리즘 A가 존재한다면, 먼저 A를 사용하여 여러 개의 가능한 해들을 생성한 다음 이를 초기의 개체군으로 생각하고 그 위에 진화 알고리즘을 적용함으로써 적어도 기존의 알고리즘 A의 결과 보다는 더 우수한 결과를 얻을 수 있다. 진화 알고리즘이 흥미를 끄는 또 하나의 이유는 계산 모델이 자연 현상에 기반을 두고 있다는 것이다. 인간이 오랜동안 풀려고 했던 많은 실세계의 문제들을 자연은 실제로 유연하게 해결하고 있다.

진화 알고리즘이 기존의 방법들보다 더 효과적으로 적용될 수 있는 응용 영역의 성격을 정확히 규정하는 문제는 아직 더 연구 되어야 할 과제로 남아 있으나, 일반적으로 다음과 같은 특성을 갖는 문제에 대해서는 진화 계산 방법이 효과적으로 적용됨이 많은 응용 결과에 의하여 확인되었다.

- 여러 개의 국부적 해가 존재하는 문제 (multi-modal)
- 어느 정도의 규칙성을 가지고 있는 문제 (regularity)
- 문제 영역의 규칙성이 어느 정도 염색체로 표현가능한 문제
- 부분적인 해들 간에 상대적인 우위관계가 (epistatis) 존재하는 문제

현재 추세로 간다면 유전 알고리즘의 응용 범위는 점점 더 늘어날 것이며 이론적인 발전도 가속화되리라 예상된다. 특히 지능 제어와 관련하여 진화 계산 방식은 신경망과 퍼지로직과 결합되어 적응 학습 및 최적화 문제의 새로운 해결법으로 많은 새로운 결과들이 나올 것으로 예상된다.

참 고 문 헌

[1] 「Baeck & Schwefel 1993」 T. Baeck and H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evolutionary Computation*, 1(1) : 1-23, 1993.

[2] 「Carse et al. 1995」 B. Carse, T. C. Fogarty and A. Munro, Adaptive distributed routing using evolutionary fuzzy control, in *Proc. of 6th Int. Conf. on Genetic Algorithms*, L. J. Eshelman (ed.), Morgan Kaufmann, San Francisco, C.A., 1995, pp. 389-396.

[3] 「Fogel 1966」 D.B. Fogel, *Artificial Intelligence through Simulated Evolution*, John Wiley, N.Y., 1966.

[4] 「Fogel 1995」 D.B. Fogel, *Evolutionary Computation : Toward a New Philosophy of Machine Intelligence*, IEEE Press, 1995.

[5] 「Goldberg 1989」 D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

[6] 「Goldberg 1994」 D. E. Goldberg, Genetic and evolutionary algorithms come of age, *Communications of the ACM*, 39(3) : 113-119, 1994.

[7] 「Holland 1975」 J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, M. I., 1975.

[8] 「Koza 1992」 J. R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.

[9] 「Muehlenbein 1993」 H. Muehlenbein and D. Schlierkamp-Voosen, Predictive models for the breeder genetic algorithm I : Continuous parameter optimization, *Evolutionary Computation*, 1(1) : 25-49, 1993.

[10] 「Rechenberg 1973」 I. Rechenberg, *Evolutionsstrategie : Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution*, Frommann-Holzberg Verlag, Stuttgart, 1973.

[11] 「Schaffer et al. 1992」 J. D. Schaffer, D. Whitley, and L. J. Eshelman, Combinations of genetic algorithms and neural networks : A survey of the state of the art, in *Proc. Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, IEEE, 1992, pp. 1-37.

[12] 「Srinivas & Patnaik 1994」 M. Srinivas and L. M. Patnaik, Genetic algorithms : a survey, *IEEE Computer*, June 1994, pp. 17-26.

[13] 「Zhang 1992」 B. T. Zhang, *Lernen durch Genetisch-Neuronale Evolution*, DISKI Vol. 16, ISBN 3-929037-16-5, Infix-Verlag, Bonn/St. Augustin, 1992.

[14] 「Zhang & Muehlenbein 1993」 B. T. Zhang and H. Muehlenbein, Evolving optimal neural networks using genetic algorithms with Occam's razor, *Complex Systems*, 7(3) : 199-220, 1993.

[15] 「Zhang 1994」 B. T. Zhang, Accelerated learning by active example selection, *International Journal of Neural Systems*, 5(1) : 67-75, 1994.

[16] 「Zhang & Muehlenbein 1995」 B. T. Zhang and H. Muehlenbein, Balancing accuracy and parsimony in genetic programming, *Evolutionary Computation*, 3(1) : 17-38, 1995.

[17] 「Zhang & Muehlenbein 1996」 B. T. Zhang and H. Muehlenbein, Adaptive fitness functions for dynamic growing/pruning of program trees, *Advances in Genetic Programming 2*, MIT Press, Cambridge, M. A., Chapter 13, 1996.

저 자 소 개

張 炳 卓

1963年 7月 11日生
1986年 2月 서울대학교 컴퓨터공학과 학사
1988年 2月 서울대학교 대학원 컴퓨터공학과 석사
1992年 7月 독일 Bonn대학교 전산학과 박사

1988年 10月～1992年 8月 Universitaet Bonn, AI Lab. 연구원
1992年 9月～1995年 8月 독일국립전산학 연구소(GMD) 연구원
1995年 9月～현재 건국대학교 컴퓨터공학과 조교수

주관심분야 : 진화계산, 신경망시스템, 기계학습, 인공지능