강화학습을 이용한 진화 알고리즘의 성능개선에 대한 연구

A Study on Performance Improvement of Evolutionary Algorithms Using Reinforcement Learning

이 상 환, 심 귀 보

로보틱스 및 지능정보 시스템 연구실
중앙대학교 공과대학 전자전기공학부
Tel : 02)820-5319, Fax : 02)817-0553, E-mail : kbsim@cau.ac.kr

Sang-Hwan Lee, Kwee-Bo Sim

Robotics & Intelligent Information System Lab.
School of Electrical & Electronic Eng., Chung-Ang Univ.
Tel : +82-2-820-5319, Fax : +82-2-817-0553, E-mail : kbsim@cau.ac.kr

ABSTRACT

Evolutionary algorithms are probabilistic optimization algorithms based on the model of natural evolution. Recently the efforts to improve the performance of evolutionary algorithms have been made extensively. In this paper, we introduce the research for improving the convergence rate and search faculty of evolution algorithms by using reinforcement learning. After providing an introduction to evolution algorithms and reinforcement learning, we present adaptive genetic algorithms, reinforcement genetic programming, and reinforcement evolution strategies which are combined with reinforcement learning. Adaptive genetic algorithms generate mutation probabilities of each locus by interacting with the environment according to reinforcement learning. Reinforcement genetic programming executes crossover and mutation operations based on reinforcement and inhibition mechanism of reinforcement learning. Reinforcement evolution strategies use the variances of fitness occurred by mutation to make the reinforcement signals which estimate and control the step length.

Ⅰ. 서 론

진화 알고리즘(evolutionary algorithms ; EAs)[1,2]은 생물의 진화과정을 모델링하여 복잡한 문제를 해결하고자 하는 계산모델이다. 이는 개체군내에서의 집단적 탐색과정에 기초를 두고 있으며, 각 개체는 탐색공간내의 하나의 탐색점에 대응된다. 임의로 형성된 초기 개체군은 세대가 지날수록 연산과 선택과정을 통해 탐색공간내에서 적합도가 좋아지는 방향으로 진화해 나간다. 진화 알고리즘의 네 가지 주요 분야로는 유전자 알고리즘(genetic algorithms ; GAs), 진화 전략(evolution strategies ; ESs), 진화 프로그래밍(evolutionary programming ; EP), 유전자 프로그래밍(genetic programming ; GP) 등이 있다.

그러나 진화 알고리즘은 아직도 자연계의 현상들을 정확히 모델링하였다고 보기 어려울 뿐만 아니라, 적용되는 문제들이 복잡해짐에 따라 진화의 속도나 탐색능력의 개선이 요구되고 있다. 이러한 한계들을 극복하고자 다각적인 연구가 활발히 진행되고 있다.

한편, 강화학습(reinforcement learning)[3]은 고등 동물의 학습 방법 연구에서 비롯되었으며, 동적 환경 하에 있는 에이전트(agent)의 행동에 대한 보상을 최대화하는 상태-행동 규칙이나 행동 발생 전략을 찾는 비교사 학습법이다. 즉 보상의 최대화라고 하는 목적만을 가지고 있기 때문에 본질적으로 불확실성과 지연 보상이 존재하지만 환경으로부터의 보상과 벌칙만으로 우수한 정보를 획득할 수 있다는 장점 때문에 애초에 정답이 없는 문제에의 접근의 길을 열어주고 있다.

본 논문에서는 이러한 강화학습을 진화 알고리즘에 적용하여, 개체의 적합도의 변화를 최대화할 수 있도록 연산과정을 학습시키는 방법을 제안한다. 즉, 진화 알고리즘의 각 분야에 맞도록 적절한 강화학습을 사용하여 성능의 향상을 기하였다. 유전자 알고리즘에서는 각 유전자좌에 적용될 돌연변이 확률을 적응적으로 생성해 나갈 수 있도록 강화학습을 적용하였다. 유전자 프로그래밍에서는 환경으로부터의 보상을 통해 각 노드의 돌연변이 확률 및 교차 확률을 결정하도록 하였고, 또한 진화 전략에서는 표준편차 벡터의 돌연변이에 방향성을 제공하는데에 강화학습을 사용함으로써 각각의 성능을 향상시킬 수 있었다.

본 논문의 구성은 다음과 같다. 2장에서는 진화연산의 각 분야에 대해서, 또 3장에서는 강화학습에 대해서 그 특징을 간략히 소개한다. 4장부터 6장까지는 유전자 알고리즘, 유전자 프로그래밍, 그리고 진화 전략 각각에 대하여 강화학습을 이용한 성능개선방법을 소개한다. 7장에서는 결론을 맺는다.

Ⅱ. 진화 알고리즘

진화 알고리즘은 자연계의 진화과정을 컴퓨터 상에서 시뮬레이션함으로써 복잡한 실세계의 문제를 해결하고자 하는 계산모델이다. 2장에서는 이러한 진화 알고리즘의 각 분야에 대한 그 방법과 특징을 간략히 소개한다.

2.1. 유전자 알고리즘

유전자 알고리즘[4]은 1975년 John Holland의 연구에 기원한 것으로 개체군(population) 중에서 환경에 대한 적합도(fitness)가 높은 개체가 높은 확률로 살아남아 재생(reproduction)할 수 있게 되며, 이때 교차(crossover) 및 돌연변이(mutation)로서 다음 세대의 개체군을 형성하는 과정을 갖는다. 유전자 알고리즘은 풀고자 하는 문제의 변수값을 이진 스트링으로 표현한다. 주 연산자로는 교차를 사용하며 이는 두 개체 사이의 염색체 교환을 통하여 새로운 개체를 생성하는 방법이다. 부 연산자로는 돌연변이를 사용하며 이는 개체군의 다양성을 유지하고 개체에 근접한 새로운 개체를 생성하는 방법이다. 또한 유전자 알고리즘의 수렴성을 증명하는 이론적인 기반으로는 스키마 정리와 빌딩블록 가설이 존재한다.

유전자 알고리즘은 진화 알고리즘 중에서 가장 활발하게 여러 가지 문제에 적용되고 있으며 그 연구 또한 다양하게 이루어지고 있다. 생물학적인 DNA의 구조를 모델링한 DNA coding 방법[5]과 개체의 생존기간동안 수행되는 학습을 통해 선택에 영향을 미치는 혼성 유전자 알고리즘[6] 등이 활발히 연구되고 있다.

2.2 유전자 프로그래밍

유전자 프로그래밍[7]은 90년대 초 Stanford 대학의 John Koza에 의해 개발, 소개되어 연구가 진행중이며 아직 개발의 여지가 많이 남아 있다. 유전자 프로그래밍의 염색체 표현은 유전자 알고리즘의 염색체 표현 방법을 확장하여 LISP언어의 S-식 표현방법인 트리구조로 되어 있으며 다른 진화 알고리즘 방법과 마찬가지로 트리구조의 프로그램을 임의로 여러 개를 생성한 후 이것의 수행능력에 따라 경쟁적으로 선택, 도태되어 진화시킴으로서 원하는 프로그램을 얻어내는 방법이다. LISP 프로그래밍 언어의 Symbolic Expression은 이러한 트리구조를 갖는 유전자 프로그래밍의 함수와 말단 기호의 구성을 생성하고 조작하는데 특별히 편리한 방법이다. 유전자 프로그래밍의 각 개체는 각 문제에 대하여 그 문제의 영역에 알맞게 정해진 함수와 말단 기호들로 구성된 모든 결합체들의 무제한의 공간이다. 이 점이 고정길이 염색체 문자열을 갖는 유전자 알고리즘과 대조되는 유전자 프로그래밍의 장점이라고 할 수 있다. 유전자 알고리즘에서 스트링으로 표시되던 염색체를 트리구조의 프로그램으로 확장하면서 학습, 추론, 문제해결 등 적용분야도 넓어졌다. 프로그램 진화를 위한 기본적인 조작 방법으로는 노드를 임의적으로 변경하는 돌연변이와 서로 다른 트리의 부분트리를 교환하는 교차, 그리고 한 트리내에서 노드간의 위치를 바꾸는 역위 등이 존재한다.

2.3. 진화 전략

진화 전략[8]은 실수치 벡터를 사용하는 알고리즘으로서 함수 최적화 문제에 많이 이용되고 있다. 최초의 진화전략은 1964년 독일에서 유체역학 문제에 적용되어졌다. 1973년에 Rechenberg는 하나의 개체가 하나의 자손을 생성하는 (1+1)-ES의 수렴속도에 대한 이론으로서, gaussian 분포의 표준편차를 성공하는 돌연변이의 수에 따라 변화시키는 1/5-성공규칙(1/5-success rule)을 제안하였다. 이후 Schwefel에 의해 μ개의 부모개체와 λ개의 자손개체로부터 다시 μ개의 부모개체를 생성하는 $(\mu+\lambda)$-ES와 λ개의 자손개체로부터 μ개의 부모개체를 생성하는 (μ,λ)-ES가 제안되었다. 이러한 진화 전략의 주 연산자로는 돌연변이를 사용하며 부 연산자로 재조합(recombination)을 사용한다. 진화전략의 하나의 개체는 개체(object variables)벡터 $\vec{x}$와 표준편차(standard deviations) 벡터 $\vec{\sigma}$, 그리고 회전각(rotation angle)벡터 $\vec{\alpha}$를 갖으며 돌연변이 시에는 각각의 벡터가 식 (1)과 같이 자기적응(self-adaptation)적으로 돌연변이

된다.

$$\begin{aligned}\sigma_i' &= \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \\ \alpha_j' &= \alpha_j + \beta \cdot N_j(0,1) \\ x_i' &= x_i + z_i(0, \sigma', \alpha')\end{aligned} \quad (1)$$

여기서 $\tau' \cdot N(0,1)$는 모든 $i \in \{1, \ldots, n\}$에 대하여 동일한 gaussian 랜덤 변수값을 갖는 전역적 인수(global factor)이며 $\tau \cdot N_i(0,1)$는 매 i번째마다 새롭게 발생되는 gaussian 랜덤 변수값이 적용되는 개별적 인수(individual factor)이다. $z(0, \sigma, \alpha)$는 일반화된 n차의 gaussian 분포(the generalized n-dimensional normal distribution)에 따라 발생되는 랜덤 벡터이다.

2.4. 진화 프로그래밍

진화 프로그래밍[2]은 예측문제의 해를 구하기 위하여 유한 상태 기계(FSM : finite state machine)를 진화시키는 방법으로서 1960년대에 Fogel에 의해서 개발되었다. 이는 진화전략과 매우 유사하나, 진화전략과는 달리 교배 연산자를 사용하지 않고 확률적인 선택을 한다. 각 개체는 정규분포의 분산 $\vec{\nu} \in \mathbf{R}^n$를 개체표현의 일부로 추가하여 $(\vec{x}, \vec{\nu})$로 나타내며, 돌연변이에 의한 다음 세대의 개체는 n차원의 실수 공간상에서 다음과 같이 표현된다.

$$\begin{aligned}x_i' &= x_i + \sqrt{\nu_i} \cdot N_i(0,1) \\ \nu_i' &= \nu_i + \sqrt{\alpha \nu_i} \cdot N_i(0,1)\end{aligned} \quad (2)$$

또한 적합도 함수 $\Phi(\vec{a})$는 목적함수 $f(\vec{x})$의 스케일링과 약간의 random alternation k를 가하여 다음과 같이 된다.

$$\Phi(\vec{a}) = \delta(f(\vec{x}), k) \quad (3)$$

여기서, $\delta(\cdot)$는 스케일링 함수이다. 이것은 목적함수에 잡음을 가하여 적합도 평가에 약간의 확률적인 요소를 두기 위함이다. 진화 프로그래밍에서는 돌연변이를 통하여 μ개의 부모 개체로부터 μ개의 자손 개체를 생성한다. 이때 하나의 부모 개체는 하나의 자손 개체를 생성한다. 이후 확률적인 q-토너먼트 선택법에 의하여 부모와 자손 개체(2μ)개 중에서 μ개의 개체를 선택하여 다음 세대의 부모로 한다.

Ⅲ. 강화학습

강화학습은 실험 심리학에서 동물의 학습방법의 연구에서 비롯되었으나, 최근에는 공학에서 여러 다른 인공지능 분야들과 합성하여 학습알고리즘을 향상시키는데 많이 이용되고 있다. 가장 간단한 강화학급 방법은 어떤 행동을 했을 때 수행하고 있는 작업의 방향으로 향상된 상태를 가져오면 그 행동을 유발하기 위하여 그 행동에 강화를 준다는 일반적인 상식의 생각에 기초한다.

강화학습은 일반적으로 제어기 또는 에이전트의 행동에 대한 보상을 최대화하는 상태-행동 규칙이나 행동 발생 전략을 찾는 것이다. 그러나 많은 실세계의 경우에 있어서 목표상태에 도달할 때까지는 중간 단계의 행동에 대한 즉각적인 보상이 주어지지 않는다. 이러한 경우 외부로부터의 강화 신호가 없기 때문에 학습이 일어나지 않게 된다. 그러한 경우에도 목표상태에 도달하기 위해서는 계속적인 학습이 이루어져야 하므로 일시적인 신뢰 할당이 이루어져야 한다. 이것을 신뢰 할당 문제(credit-assignment problem)라고 하며 강화 학습에 있어서 가장 중요한 문제라고 할 수 있으며, 이 문제에 대한 가장 일반적인 접근 방법은 강화 신호를 생성하는 외부 평가 함수보다 더 자세한 정보를 얻을 수 있는 내부 평가 함수를 구현하는 것이다. 이러한 강화학습의 기본구조를 그림 1에 나타내었다.

한편, 강화학습법은 현재까지 Sutton의 TD method에 의한 actor-critic 구조[9]와 Watkins의 Q-learning[10] 등이 개발되어 있다. 그러나 이것은 마코비언 환경(markovian environment)의 환경동정형(exploration oriented) 학습방법으로 앞으로는 비마코비언 환경의 경험강화형(exploitation oriented)의 학습방식으로 발전해 갈 전망이다.

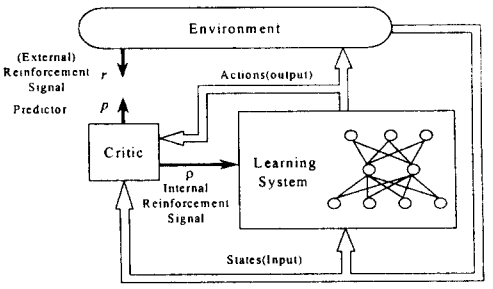


그림 1. 강화학습의 기본 구조

Ⅳ. 적응 유전자 알고리즘

유전자 알고리즘에서는 하나의 개체가 연산과정을 거치면서 그 개체가 갖고 있던 적합도에 변화가 발생한다. 이러한 적합도의 변화는 환경으로부터의 보상(reward)으로 볼 수 있다. 적응 유전자 알고리즘은 이 보상이 최대가 되도록, 즉, 적합도가 보다 좋아지도록 돌연변이

의 파라메터를 결정한다. 따라서, 강화학습에 있어서의 상태(state)는 연산과정 이전의 유전자형(genotype) 그 자체로, 행동(action)은 연산과정으로 각각 대응될 수 있다. 적응 유전자 알고리즘(adaptive genetic algorithms)[11]은 유전자 알고리즘의 연산 중에서 돌연변이 연산에 강화학습을 적용하였다. 이를 위해 다수의 상태에 대해서도 처리가 용이한 신경망을 도입하였다. 따라서, 유전자 알고리즘의 진화과정 중에서 돌연변이 시에는 신경망의 출력값을 이용하여 돌연변이를 수행한다. 그 결과인 적합도 변화를 이용하여 신경망을 학습하고 다시 유전자 알고리즘을 반복 수행하는 과정을 거치게 된다. 그림 2에 적응 유전자 알고리즘의 구조를 나타내었다.

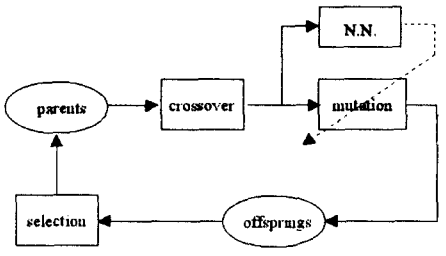


그림 2. 적응 유전자 알고리즘의 구조

신경망의 입력은 각 개체의 유전자형으로, 출력은 각 유전자좌(locus)의 돌연변이 확률로 설정한다. 현재의 개체 x_t가 돌연변이를 통해 받는 보상 $r(x_t)$는 식 (4)와 같이 돌연변이 이후의 적합도 $f(x_{t+1})$와 돌연변이 이전의 적합도 $f(x_t)$사이의 차이로 정의된다.

$$r(x_t) = f(x_{t+1}) - f(x_t) \tag{4}$$

또, 교사신호의 유도를 위해서, 실제로 해당 유전자좌에서 돌연변이가 일어났는가의 값을 갖는 변수가 필요하다. 이 변수 c_i는 i번째 유전자좌에서 돌연변이가 일어나면 1을, 돌연변이가 일어나지 않으면 0을 갖는다. 위에서 언급한 보상 $r(x_t)$와 변수 c_i를 이용하여 i번째 유전자좌의 원하는 돌연변이 확률 즉, i번째 출력뉴런의 원하는 출력값 y_i^*은 다음 식 (5)와 같이 정의된다.

$$y_i^* = \begin{cases} y_i + \alpha r(x_t)(c_i - y_i) & \text{if } r(x_t) \geq 0 \\ y_i + \alpha c_i r(x_t)(y_i - \overline{c_i}) & \text{if } r(x_t) < 0 \end{cases} \tag{5}$$

위 식 (5)는 다음과 같은 의미를 갖는다. 즉, 하나의 개체가 x_t라는 상태에서 c_i라는 행동을 행하여 그 결과가 보다 향상되었다면 ($r(x_t) \geq 0$), 실제 행한 행동 c_i는 더욱 강화될 수 있도록 신경망의 교사신호로서의 의미를 부여한다. 또한 반대의 경우에는 실제 행한 행동은 더욱 약화되도록 한다. 결국 신경망의 i번째 출력뉴런의 오차 추정값은 식 (6)과 같다.

$$e_i = y_i^* - y_i \tag{6}$$

이 출력오차 추정값을 역전파 알고리즘에 적용하여 학습을 수행함으로서 신경망은 성공적인 돌연변이를 유도할 수 있도록 학습하게 된다.

V. 강화 유전자 프로그래밍

일반적인 유전자 프로그래밍에서는 각 노드의 교차 및 돌연변이의 확률이 일정하다. 따라서 모든 노드는 랜덤하게 선택되어 교차 및 돌연변이의 과정을 거친다. 그러나 이것은 유용한 스키마(빌딩 블록 : building block)에 대하여도 똑같은 비율로 연산이 적용되므로 유용한 스키마가 손실될 가능성도 많으며 이것은 성능의 저하로 이어진다. 강화 유전자 프로그래밍(reinforcement genetic programming)[12]은 이와 같은 유용한 스키마가 손실될 가능성을 줄이기 위하여 강화신호를 이용해 돌연변이 지점과 교차 지점을 결정한다. 그러나 이러한 방법은 현재로서는 로봇의 제어와 같은 하향식으로 실행되는 문제에만 적용할 수 있으며 함수근사나 패리티 문제와 같은 상향식으로 실행되는 문제에 대하여는 적용하기 어렵다. 강화 유전자 프로그래밍은 로봇의 프로그램을 진화시키는 것을 목적으로 하향식으로 실행 문제에 대하여만 생각한다.

로봇의 제어와 같은 문제의 종단기호집합(terminal set)은 행동(action) 또는 센서 입력값(sensor input)이 된다. 또한 행동은 환경에 대하여 상호작용 하는 출력으로서 행동에 대한 보답을 구할 수 있다. 따라서 종단기호집합 중 행동에 대한 강화신호를 계산할 수 있으며 이 값을 근거로 부모노드(함수 집합 : function set)에 대한 강화값을 유추할 수 있다. 각 종단기호가 각각 N_i회씩 실행되고 누적된 강화값이 그림 3-(a)와 같이 주어졌을 때 돌연변이 확률 계산을 위한 각 노드에 대한 강화값(rm)의 계산식은 (7)식과 같으며, 그 결과는 그림 3-(b)와 같다.

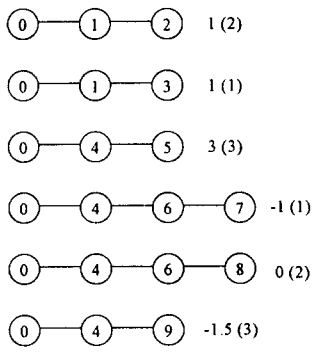
$$rm_i = \frac{\sum_{n \in T} ram_n}{\sum_{j \in T} N_j} \tag{7}$$

단, T는 부분트리(subtree)에 속하는 종단기호의 노드 집합, N_j는 노드 j의 실행 회수,

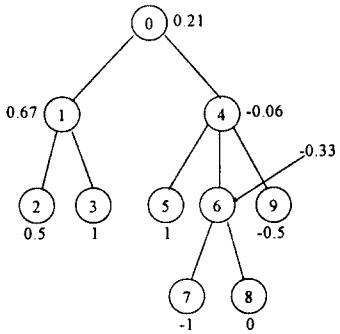
ram_n은 노드 n의 누적된 강화값, rm_i는 노드 i의 강화값이다.

예를 들어 4번 노드의 강화값은 (7)식에 의해 다음과 같이 계산된다.

$$rm_4 = \frac{3+(-1)+0+(-1.5)}{3+1+2+3} = -0.06 \quad (8)$$



(a) 각 종단기호의 누적된 강화값 및 실행된 회수



(b) 각 노드의 강화값

그림 3. 돌연변이 확률 계산을 위한 노드의 강화값

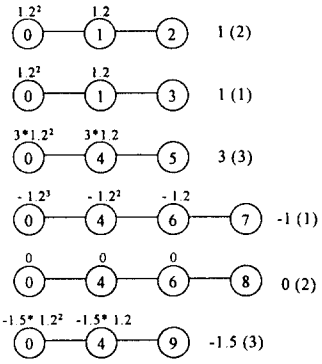
또한 강화신호값이 클수록(보상 : reward) 돌연변이 확률을 작게 하고 값이 작을수록(벌칙 : penalty) 돌연변이 확률을 크게 하기 위하여 다음과 같은 식을 사용하였다.

$$pm_i = e^{-2rm_i} pm \quad (9)$$

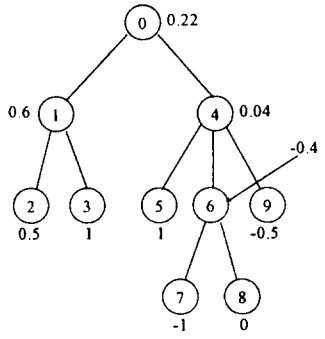
단, pm_i는 노드 i의 돌연변이율, pm은 돌연변이율의 기본값이다. 위 (9)식에서 노드 i의 강화값 rm_i가 0일 때 기본적인 pm을 적용받으며 -1일 때 e^2pm, 1일 때 $e^{-2}pm$의 돌연변이율을 적용받는다.

교차확률의 계산은 돌연변이율에 비하여 약간 복잡하다. 그 이유는 교차연산에 의하여 훌륭한 스키마(building block)의 파괴를 최소화하고 나쁜 부분트리를 제거하기 위함인데 그 방법은 다음과 같다.

한 노드의 강화값이 크다는 것은 부분트리에 속하는 종단기호가 모두 좋은 강화신호를 받았다는 것이 된다. 또 반대로 강화값이 작을 경우는 자식트리의 종단기호가 모두 안 좋은 강화신호를 받은 것이 된다. 따라서 좋은 부분트리의 손실을 최소화하고 안 좋은 트리를 다른 것과 바꾸기 위하여 노드의 깊이(depth)를 고려해 교차를 위한 강화신호(rc)를 구하였다. 그림 4는 각 종단기호가 한 번씩 실행되었을 경우 각 노드의 교차점 계산을 위한 강화값을 구하는 방법을 보인 예제이며 그 식은 (10)식과 같다.



(a) 각 종단기호의 누적된 강화값 및 실행된 회수



(b) 각 노드의 강화값

그림 4. 교차점 계산을 위한 노드의 강화값(rc)(α=1.2)

교차를 위한 강화값의 설정에 있어서는 노드의 깊이(depth)에 따른 강화상수 α(α>1)를 도입하였다. 그 이유는 일반적으로 부모의 노드는 자식노드의 강화값의 대략적인 평균값을 갖기 때문에 보상과 벌칙을 받은 자식노드가 함께 존재할 경우 부모노드의 강화값이 0에 가깝게 된다. 따라서 교차점은 주로 종단노드의 벌칙을 많이 받은 노드에서 실행될 가능성이 높다. 이

것은 교차에 의해 새로 생성되는 프로그램의 길이를 극단적으로 짧게 하거나 길게 하기 때문에 교차의 성능을 떨어뜨린다. 따라서 부모의 노드는 작은 값에서도 교차지점으로 선택될 확률이 높은 것이 바람직하다.

$$rc_i = \frac{\sum_{n \in T} \alpha^{(d_n - d_i)} \times rac_n}{\sum_{j \in T} N_j} \quad (10)$$

단, rac_n은 노드 n의 누적된 강화값, rc_i는 노드 i의 강화값, N_j는 노드 j의 실행 회수, α는 강화상수, d_i, d_n은 각각 노드 i와 n의 깊이(depth)이다.

이때, 교차점의 선택을 위한 비율값은 강화값에 대한 시그모이드 함수의 형태로 (11)식과 같이 정하였다.

$$cv_i = \frac{1}{2}(1 - \tanh(2rc_i)) \quad (11)$$

단, cv_i는 교차점 선택비율값이다. 이때, 임의의 노드 i가 교차점으로 선택될 확률 $P(s_i)$은 cv_i값에 비례하여 (12)식과 같이 구해진다.

$$P(s_i) = \frac{cv_i}{\sum_{n=0}^{N-1} cv_n} \quad (12)$$

단, N은 총 노드의 개수이다. 역위의 경우 교차의 강화값 rc_i를 그대로 이용해 (13)식과 같이 구할 수 있다.

$$pi_i = (1 - \tanh(2rc_i))pi \quad (13)$$

단, pi_i는 노드 i의 역위확률, pi는 역위확률의 기본값이다.

한편, 종단노드의 강화신호 계산은 간단한 문제일 경우 주어진 상황을 고려해 계산해줄 수 있으나, 문제에 따라 보상과 벌칙의 값이 몇 개의 값으로 주어지지 않고 복잡할 경우 퍼지추론에 의해 계산하는 방법도 있다

VI. 강화 진화 전략

강화 진화 전략(reinforcement evolution strategies)[13]은 일반적인 진화 전략에 강화학습을 이용한 진화 전략이다. 여기서 일반적인 진화 전략이란 재조합과 회전각벡터를 사용하지 않는 진화 전략으로 한정한다.

강화 진화 전략은 강화학습과는 주어진 환경에서의 차이가 존재한다. 먼저 강화학습에서는 일반적으로 하나의 에이전트가 학습하는 과정이지만 진화 전략에서는 다수의 개체가 진화하는 과정이다. 또한 연산을 통한 개체의 적합도 변화를 보상으로 보았을 때 진화 전략에서는 보상(reward)을 많이 받는 개체는 살아남고, 벌칙(penalty)을 많이 받는 개체는 도태되어 다음 학습에 반영되지 못하는 현상이 발생한다. 결국 시행착오를 통한 경험은 보상을 많이 받는 개체에만 남아 있게 된다. 또한 강화 진화 전략에서의 보상은 연산을 통한 개체의 적합도 변화로 볼 수 있으나 일반적인 강화학습과는 달리 지연되지 않고 연산과정이후에 바로 얻어진다. 이러한 차이점 때문에 일반적인 강화학습 알고리즘을 변형하여 적용할 필요성이 발생한다.

개체가 매세대마다 받는 일시적인 보상 $r(t)$은 식 (14)과 같이 정의된다.

$$r(t) = \begin{cases} +0.5 & \text{if } \Delta f(t) > 0 \\ 0 & \text{if } \Delta f(t) = 0 \\ -1 & \text{if } \Delta f(t) < 0 \end{cases} \quad (14)$$

여기서, $\Delta f(t)$는 돌연변이 이전의 적합도와 돌연변이 이후의 적합도 차이로서 다음과 같다.

$$\Delta f(t) = f(t) - f(t-1) \quad (15)$$

돌연변이 연산시에는 이러한 일시적인 보상을 n세대동안 총합한 $r_{sum}(t)$를 이용한다.

$$r_{sum}(t) = \frac{1}{n} \sum_{i=0}^{n-1} r(t-i) \quad (16)$$

즉, 돌연변이 이후의 개체의 적합도 $f(t)$가 n세대동안 연속해서 높아졌다면 총 +0.5의 보상을, 연속해서 낮아졌다면 총 -1의 벌칙을 받는다. 보상과 벌칙의 불균형은 보상을 많이 받는 개체가 전체 개체군의 주류를 이루는데에 기인한다. 강화 진화 전략은 개체의 보상을 이용하여 돌연변이의 탐색폭을 제어한다. 일반적으로 돌연변이의 탐색폭은 개체의 표준편차벡터에 의해 결정된다. 진화 전략에서는 이러한 표준편차벡터를 자기적응적으로 탐색하나, 강화 진화 전략에서는 개체의 보상이 최대화되도록 표준편차벡터에 방향성을 제공한다. 즉, 개체의 보상이 클수록 탐색폭을 크게 하며, 보상이 작을수록 탐색폭을 작게 한다. 결국 강화 진화 전략에서의 표준편차벡터는 다음과 같이 돌연변이된다.

$$\sigma_i' = \sigma_i \cdot \exp(r_{sum}(t) \cdot |\tau' N(0,1) + \tau N_i(0,1)|) \quad (17)$$

일반적인 진화 전략은 자기적응에 의해 세대가 지날수록 표준편차 벡터값이 줄어드는 경향을 갖는다. 그러나 지나치게 줄어든 표준편차는 적합도가 낮은 개체임에도 불구하고 돌연변이의 탐색폭을 줄이는 효과를 갖는다. 이는 수렴

속도와 전역적 탐색능력을 악화시키는 원인이 된다. 강화 진화 전략에서는 보상이 클수록 탐색폭을 크게 한다. 다음 세대의 부모 개체는 대부분의 경우 적합도가 높아진 개체로 구성된다. 따라서 강화 진화 전략은 지속적으로 탐색폭을 증가시킨다. 또한 지나치게 큰 표준편차를 가진 개체는 선택과정에 의해 도태되어 과도한 탐색폭의 증가를 억제한다. 결국 보상에 의한 탐색폭의 증가와 선택에 의한 탐색폭의 억제로 인해 적절한 탐색폭을 유지해간다. 이와 같이 방향성을 가하는 방법은 진화 전략의 수렴속도를 향상시키는 효과를 갖으나 전역적 탐색능력을 보장하지는 않는다. 일반적으로 수렴속도와 전역적 탐색능력은 상반되는 특성을 갖기 때문이다. 강화 진화 전략은 전역적 탐색능력의 향상을 위해 식 (18)과 같이 개체값의 돌연변이시 gaussian 분포 대신에 cauchy 분포를 사용한다.

$$x_i' = x_i + \sigma_i' \delta_t \qquad (18)$$

여기서 δ_t는 스케일 파라메터(scale parameter) $t=1$을 갖는 cauchy 랜덤 변수이다. 평균이 0인 1차의 cauchy 분포 함수는 다음과 같다.

$$f_t(x) = \frac{1}{\pi} \frac{t}{t^2 + x^2}, \quad -\infty < x < \infty \qquad (19)$$

cauchy 분포는 gaussian 분포보다 넓은 폭을 갖기 때문에 지역 최소값에서 벗어날 확률이 높아진다. 강화 진화 전략은 강화학습을 통해 수렴속도를 향상시키고, cauchy 분포를 통해 전역적 탐색능력을 향상시키는 효과를 갖는다.

Ⅶ. 결론

인공생명 연구의 한 주요 분야를 차지하고 있는 진화 알고리즘은 그 자체가 자연계의 진화현상을 모델링하고 있지만 마치 진화과정을 거치듯 지속적인 발전을 거듭하고 있다. 앞으로도 진화 알고리즘의 성능개선을 위한 연구에 많은 노력들이 기울여질 것이 분명하다. 본 논문에서는 진화 알고리즘과 강화학습을 간략히 소개하고 두 알고리즘 사이의 관계를 유추하였다. 또한 진화 알고리즘의 성능개선을 위한 연구의 한가지로서, 확률에 의존하는 진화 알고리즘에 강화학습을 적용하여 연산의 파라메터를 자동생성하고 연산에 방향성을 제공하는 방법을 소개하였다. 유전자 알고리즘에서는 각 유전자좌에 적용될 돌연변이 확률을 적응적으로 생성해나갈 수 있도록 강화학습을 적용하였다. 유전자 프로그래밍에서는 환경으로부터의 보상을 통해 각 노드의 돌연변이 확률 및 교차 확률을 결정하도록 하였고, 또한 진화 전략에서는 표준편차 벡터의 돌연변이에 방향성을 제공하는데에 강화학습을 사용함으로써 각각의 성능을 향상시킬 수 있었다.

감사의 글

이 논문은 1997년 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

참고 문헌

[1] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 1-23, 1993.

[2] D. B. Fogel, *Evolutionary Computation*, IEEE Press, 1995.

[3] J. S. R. Jang, C. T. Sun, and E. Mizutani, *Neuro-Fuzzy and Soft Computing*, Prentice Hall, 1997.

[4] M. Mitchell, *An Introduction to Genetic Algorithms*, The MIT Press, 1997.

[5] T. Yomohiro, T. Furuhashi, and Y. Uchikawa, "A Combination of DNA Coding Method with Pseudo-Bacterial GA for Acquisition of Fuzzy Control Rules," *Proceedings of 1st Online Workshop on Soft Computing*, Aug. 19-30, 1996.

[6] D. H. Ackley and M. Littman, "Interactions between learning and evolution," In, *Proc. of the 2nd Conf. on Artificial Life*, C. G. Langton, ed., Addison-Wesley, 1991.

[7] John R. Koza, *Genetic Programming: On The Programming of Computers by Means of Natural Selection*, The MIT Press, 1992.

[8] H.-P. Schwefel, *Evolution and Optimum Seeking*, A Wiley-Interscience Publication, 1995.

[9] R.S. Sutton, "Learning to Predict by the Methods of Temporal Differences, *Machine Learning*, vol 8, pp. 9-44, 1992.

[10] C.J.C.H Watkins, P. Dayan, "Technical Note : Q-Learning," *Machine Learning*, vol. 8, pp. 279-292, 1992.

[11] 이상환, 전효병, 심귀보, "강화학습을 통한 유전자 알고리즘의 성능개선," *한국퍼지 및 지능시스템학회 학술발표논문집*, 제8권, 제1호, pp. 81-84, 1998.

[12] 전효병, 이동욱, 심귀보, "강화학습에 의한 유전자 프로그래밍의 성능개선," *한국퍼지 및 지능 시스템학회 논문지*, Vol. 8, No. 3, pp. 1-8, 1998.

[13] 이상환, 심귀보, "강화학습을 이용한 진화 전략," *제13회 한국자동제어학술회의 논문집*, 제1권, pp. 326-329, 1998.