論文2000-37CI-5-3

진화와 학습의 상호 적응에 의한 자발적 주행 로봇을 위한 재귀 신경망 제어기 설계

(A Design of the Recurrent NN Controller for Autonomous Mobil Robot by Coadaptation of Evolution and Learning)

金 大 鎭 *, 姜 大 星 **

(Daijin Kim and Dae-Seong Kang)

요 약

본 논문은 장애물 회피 능력을 갖는 자발적 주행 로봇 (Khepera)을 제어하는 재귀 신경망을 진화와 학습의 상호 적응에 의해 결정하는 방안을 제시한다. 제안한 동시 적응 방안은 다음 두 가지 성질을 갖는다. 유전자 알고리즘에 의해 해집단내 여러 개의 신경망 제어기들은 전역적 탐색을 수행하여 점진적으로 장애물과의 충돌이 적게 일어나도록 진화되고, 동시에 각 신경망 제어기는 상보적 재강화 역전파 (CRBP : Complementary Reinforcement Backpropagation) 학습에 의해 국부적 탐색을 수행하여 주행 특성이 로봇이 처한 외부 환경에 적응되어진다. 실험 결과, 학습과 결합한 진화에 의해 얻어진 신경망 제어기가 진화 자체만에 의해 얻어진 신경망 제어기보다 더 나은 충돌 회피 능력을 보여 주며, 원하는 주행 성능에 보다 빨리 도달하는 것을 확인할 수 있다.

Abstract

This paper proposes how the recurrent neural network controller for a Khepera mobile robot with an obstacle avoiding ability can be determined by co-adaptation of the evolution and learning. The proposed co-adaptation scheme consists of two folds: a population of NN controllers are evolved by the genetic algorithm so that the degree of obstacle avoidance might be reduced through the global searching and each NN controller is trained by CRBP learning so that the running behavior is adapted to its outer environment through the local searching. Experimental results shows that the NN controller coadapted by evolution and learning outperforms its non-learning equivalent evolved by only genetic algorithm in both the ability of obstacle avoidance and the convergence speed reaching to the required running behavior.

* 正會員, 浦港工科大學校 컴퓨터工學科
(Dept. of Computer Science and Engineering, POSTECH)
** 正會員, 東亞大學校 電氣電子컴퓨터工學部
(School of Electrical, Electronic and Computer Engineering, Dong University)
接受日字:1998年12月29日, 수정완료일:2000年4月24日

Ⅰ. 서 론

자율적 대리자는 외부의 감시나 제어 없이도 환경 특성에 강인하고 확실한 자기 적응력을 지니고 있다[1]. 이러한 적응 능력은 대리자가 자신의 환경과의 상호 작용으로부터 창발된다. 주행 로봇으로 하여금 실제 환경 하에서 새로이 적응된 행동 양식을 스스로 계발할

수 있도록 하는 이러한 자율성을 갖도록 하는 것은 바람직하다. 자율적 주행 로봇을 제어하는데 흔히 신경망 제어기가 사용되는데 이는 신경망이 실세계에서 자율성이 요구하는 다음과 같은 여러 특징을 지니고 있기 때문이다[2]. 신경망은 첫째 신경망은 학습 능력을 통해 동적인 환경 변화에 로봇의 행동을 적응시켜주며, 둘째 일부 가중치가 없어지거나 고장나더라도 로봇의 행동에 큰 영향을 미치지 않으며, 셋째 주어진 제어 목적을 잘 수행하고, 로봇이 갖는 감지-구동 특징을 가장 잘 이용되도록 자기 자신의 구조를 변형시키며, 넷째 불가피한 내부 잡음을 갖는 물리적 센서와 엑츄에이터를 조정하는 잡음 허용적이며, 마지막으로 재귀 구조 및 lateral 연결을 통해 로봇 제어가 갖는 시간적 종속성을 처리할 수 있게 하기 때문이다.

이러한 자율적 시스템을 구축하는 한 방법으로 많은 연구자들은 자율적 로봇의 신경망 제어기를 진화에 의해 얻고자 한다[3-4]. 이 방법에 의하면 로봇 제어기의 기본 구조 및 기능을 인공적인 chromosome 형태로 나타내고, 서로 다른 특성을 나타내는 chromosome들을 집단적으로 생성시킨 후, 모든 chromosome이 갖는 제어 성능을 테스트해 본 다음, 정해진 생존 기준에 의해 만들어진 목적 함수를 얼마나 잘 만족하는가의 여부에 의해 선택 재생된다. 이러한 과정은 전체 평균 목적 함수값이 정해진 기준 이상이 되거나 또는 집단내 개체 중 어느 하나가 창발적 특성을 나타내면 멈추게 된다. 자율적 로봇의 제어기를 얻는데 이러한 진화 방법을 사용한 예는 아주 많다. 대표적 예가 D. Floreano와 F. Mondada[4]는 실제 주행 로봇(Khepera)을 제어하는 재귀 신경망 제어기를 사람의 간섭 없이 유전자 알고리즘을 사용하여 진화시켜 귀향 주행 (homing navigation) 능력을 갖도록 하였다.

자율적 로봇의 신경망 제어기의 가중치를 외부 환경 변화에 적응시키는 또 다른 방법은 재강화 학습 (reinforced learning)을 이용하는 것이다. 로봇 제어는 제어 목적이 추상적으로 표현되는 것이 대부분이어서 매번 변하는 환경 하에서 매순간 상황에 가장 적절한 동작을 결정하는 것이 불가능하다. 따라서, 재강화 학습에서는 로봇이 갖는 모든 상황에서 각기 다른 동작을 취할 수 있지만 그중 가장 유용한 동작을 취하도록 하기 위해, 주어진 상황에서 취한 동작이 제어 목적에 얼마나 유용한가를 나타내는 재강화 (또는 성능) 궤환 신호값에 의해 이 값이 양수 값이면 상을 주어 현재의 제어 방향을 유지하고, 음수 값이면 벌을 가해 현재의 제어 방향을 막는다. 자율적 로봇의 제어기를 얻는데 이러한 재강화 학습을 사용한 예로서 J. Millán[5]은 재강화 학습에 의해 단기간에 Nomad 200 주행 로봇의 주행 및 충돌 회피 능력을 갖는 여러 개의 모듈로 구성된 2층 신경망 제어기를 얻을 수 있음을 보였다.

최근, 생물학적 실험 결과에서 얻은 개념을 바탕으로 진화와 학습을 결합하는 방안이 많은 주목을 끌고 있다. 진화는 해집단 수준에서 유전자형을 변화시키는 반면, 학습은 개체 수준에서 개체의 행동 양식을 변화시킨다. 진화에 의한 변화는 개체의 선택적인 재생산성 및 변이성을 유지하기 위한 유전 연산(재결합 and/or 변이)을 수행함으로써 얻어진다. 진화에 의한 변화는 특정 한 세대에서 일어난 변화가 그 이전 세대에서 생긴 다른 변화들에 중첩되어 세대가 지날수록 누적된다. 학습에 의한 변화는 일생 동안의 특정 환경과의 상호작용과 경험을 통해 그 내부 구조에 환경의 영향을 반영함으로써 얻어진다. 학습에 의한 변화는 한 개체의 일생 중 특정 시점에서 일어난 변화가 그 이전 시점에 일어난 변화에 의해 영향을 받아 개체적으로 누적된다.

진화와 학습이 두 가지 다른 종류의 수준(각각 해집단과 개체수준)에서 일어나는 서로 다른 종류의 변화이지만, 서로 영향을 주고받는다. 진화가 학습에 미치는 영향은 진화적 변화가 유전자형(genotype)에 그 흔적을 남기고 한 개체의 게놈은 표현형(phenotype) 특성을 부분적으로 결정한다는 사실로부터 쉽게 이해할 수 있고, 그러한 영향은 개체가 앞으로 어떻게 행동할 것인가와 무엇을 학습할 것인가를 결정짓는다. 진화가 학습에 미치는 영향은 유전 알고리즘을 신경망의 해집단에 적용하는 많은 실험에서 발견할 수 있다. 진화는 신경망이 특정 과제를 학습하려는 경향을 가지도록 선택한다. 특정 과제를 학습하려는 경향은 여러 다양한 방법으로 신경망에 결합되어질 수 있다. 예를 들어, 진화는 좀 더 나은 학습이 이루어 질 수 있도록 초기 가중치 벡터[6]나 망 구조[7]를 선택할 수 있다.

학습이 진화에 어떻게 영향을 미치는 가에는 두 가지 가설이 있다. 하나는 한 생명체가 일생 동안 얻은 표현형의 특성이 유전될 수 있는 유전자형에 복사되어 한 생명체의 자손에 직접 전달된다는 Lamarckian 가설[8]이다. D. Ackley와 M. Littman[9]은 Lamarckian 가설이 컴퓨터 상에 쉽게 구현될 수 있고, 동시에 그것이 갖는 빠른 수렴 특성으로 인해 최적화 문제에 훨씬 더

효율적이라는 것을 보였다. 다른 하나는 일생 동안 얻어진 특성이 그의 자손에 직접 전달되지는 않지만 학습이 탐색 공간의 모양을 변경시키고 이로 인해 동시 적응되는 개체들의 집합에 좋은 진화 경로를 제공하는 방식으로 학습이 진화의 과정을 간접적으로 유도한다는 Baldwin 효과[10]이다. G. Hinton과 S. Nowlan[11]은 표현형에 의해 얻어진 특성들이 그의 유전자형에 직접 전달되지 않는다 하더라도 Baldwin 효과에 의해 학습하는 생명체가 학습하지 않는 것보다 더 빨리 진화한다는 것을 보여주었다. D. Parisi와 S. Nolfi[12]도 생명체가 가지는 적합성(fitness)과 간접적으로 관련이 있는 학습을 수행하는 것도 진화를 유도할 수 있음을 보여주었다. 이들의 실험 결과에 따르면, 국부해 방향으로서 기울기의 근사치가 진화를 유도하는 방향 지시 역할을 성공적으로 수행함을 보여주었다.

본 연구는 신경망 제어기를 외부 환경에 적응시키는데 재강화 신호를 받자마자 즉각 가중치를 갱신하는 국부적인 재강화 학습과 상당 시간 여러 신경망 제어기의 제어 성능을 관측한 후 이들간의 집단적인 유전 연산에 의해 진화를 하나로 결합한 새로운 상호 적응 방법을 제안한다. 제안한 상호 적응 방법은 학습에 의해 유도되는 신경망의 진화가 진화 자체만에 의한 신경망보다 더 나은 충돌 회피 능력을 보여 주며, 원하는 주행 성능에 보다 빨리 도달하는 것을 확인할 수 있음을 보인다.

본 논문은 다음과 같이 구성되어 있다. 2장은 주행 로봇 제어를 위한 재귀 신경망 제어기의 CRBP 학습과 유전 알고리즘에 의한 진화 방안을 설명한다. 3장은 학습과 진화를 결합한 상호 적응에 의한 새로운 신경망 제어기 설계 방안을 설명한다. 4장에서는 제안한 상호 적응 방법을 Khepera이동 로봇의 자발적 주행 및 충돌 회피 능력을 얻는 실험에 적용한다. 마지막으로, 결론이 뒤따른다.

Ⅱ. 신경망 제어기의 학습 및 진화

1. CRBP에 의한 신경망 제어기 학습

주행 로봇 제어 문제는 특정한 상황에서 얻어지는 센서 입력에 대한 로봇 제어기의 반응이 올바른지 아닌지를 매 순간 결정하는 것이 어려워, 가능한 충돌하지 않고 오래 주행 할 수 있어야 한다는 것처럼 추상적 목적에 의해 정의된다. 따라서, 학습시 사용되는 학습 샘플 데이터의 부재로 기존의 역전파 알고리즘을 직접 사용할 수 없다. 이를 극복하기 위해 본 연구에서는 현재의 제어 상태가 좋고 나쁜지를 나타내는 한 재강화 신호에 의해 신경망 제어기의 가중치를 갱신하는 강화 학습법의 일종인 보완적 강화 역전파 (CRBP; Complementary Reinforcement Back- propagation) 학습 알고리즘[13]을 사용하고자 한다. 여기서 '보완적'의 의미는 재강화 신호값이 크면($r>0.5$) 보상(reward)을 받아 현재 출력을 탐색점 쪽으로 움직이도록 가중치를 갱신하고, 재강화 신호값이 작으면($r<0.5$) 벌(punishment)을 받아 현재 출력을 탐색 점에서 멀어지도록 가중치를 갱신하는 것을 말한다.

본 연구에서 사용한 CRBP 학습 알고리즘은 다음과 같다. 먼저, 한 센서 입력에 대한 신경망 제어기의 순방향 연산에 의해 출력층에서의 제어 출력 O를 얻는다. 제어 출력 O 근방의 구간 $[O-\delta : O+\delta]$에서 임의의 한 출력 탐색값 S를 무작위로 발생시켜 얻는다. 여기서 δ는 탐색 구간을 나타낸다. 다음 얻어진 탐색값 S에 의해 로봇을 주행시킨다. 주행 상태 측정 결과, S가 원하는(reward) 결과를 나타내면 학습은 제어 출력 O가 탐색값 S쪽으로 움직이도록 가중치를 갱신하기 위해 오차 신호 $E=(S-O)$를 역전파시키며, S가 원치 않는(punishment) 결과를 나타내면 학습은 제어 출력 O가 탐색값 S쪽에서 멀어지도록 가중치를 갱신하기 위해

Step 1: 센서로부터 입력을 얻는다.
Step 2: 센서 입력을 신경망 제어기에 가해 제어 출력 O를 구한다.
Step 3: $[O-\delta : O+\delta]$에서 임의의 한 출력 탐색값 S를
무작위로 발생시켜 얻는다.
Step 4: 탐색값 S를 주행 로봇에 가해 재강화 신호값을 얻는다.
Step 5: 재강화 신호에 따른 오차를 결정한다.
reward이면 E = S - O
punishment이면 E = S - O
no reinforcement이면 E = 0
Step 6: E가 0이 아니면 오차를 역전파시킨다.
α_r (reward 학습율) = 0.3
α_p (punishment 학습율) = 0.1

그림 1. CRBP 학습 알고리즘의 수행 과정
Fig. 1. Procedure of CRBP learning algorithm.

오차 신호 $E=(O-S)$를 역전파 시킨다. 본 연구에서는 한 시작점에서 아래 학습 알고리즘을 한번 반복한 것을 한 번의 학습 주기로 보고, 한 시작점에 대해 1,000번의 학습 주기를 반복하고, N개의 시작점에 대해 학습을 수행하는데, 매 $1000\times N$ 학습 동안 충돌율이 30%이하이면 학습을 멈춘다. 아래 그림 1은 사용된 CRBP 학습 알고리즘의 수행 과정을 나타낸 것이다.

그림 1에서 로봇의 주행 상태가 reward인지 punishment인지는 로봇의 주행 성능을 나타내는 재강화 신호 $r=v\cdot(1-\sqrt{\Delta v})\cdot(1-i)$ $(0\le r\le 1)$값에 의해 아래 식에 의해 구분되어진다.

$$\begin{cases} reward & when\ r>0.75 \\ punishment & when\ r<0.25 \\ noreinforcement & otherwise \end{cases} \quad (1)$$

여기서, $v, \Delta v$ 와 i는 각각 [0,1] 사이의 값을 갖는 두 바퀴의 평균 속도, 두 바퀴의 속도차, 센서값중 최대 값을 나타낸다.

본 연구에서 사용된 신경망 제어기는 연속적인 값을 갖는 시그모이드 유닛으로 구성된 완전 연결형 단층 퍼셉트론 구조를 갖는데, 출력층은 지나간 시간에서의 sensory-motor 제어 정보를 간직하기 위해 출력 유닛 사이에 리커런트 연결을 갖고 있으며, 각 출력층 유닛은 입력층의 모든 유닛과 연결되어 있다. 그림 2는 본 연구에서 사용한 신경망 제어기의 구조를 나타낸 것으로 8개의 입력 유닛과 2개의 출력 유닛으로 구성된 완전 연결형 리커런트 신경망을 나타낸다. 입력 유닛 8개 중 6개는 Khepera 로봇의 전방 입력 센서에 연결되고, 2개는 Khepera 로봇의 후방 입력 센서에 연결되며, 두 개의 출력 유닛은 Khepera 로봇의 구동 모터에 직접 연결된다.

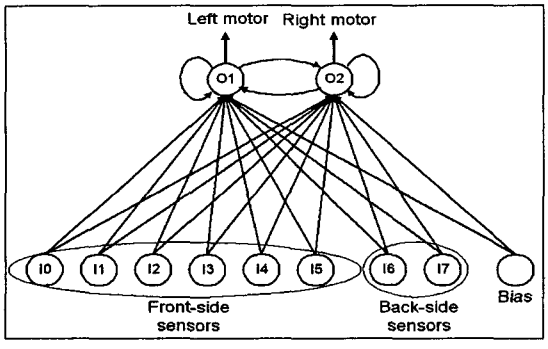

그림 2. 완전 연결형 리커런트 신경망 제어기
Fig. 2. Fully connected recurrent neural network.

사용된 리커런트 신경망 제어기의 가중치를 갱신시키는데는 William과 Zipser[14]에 의해 제안된 방법을 사용한다. 이 가중치 갱신 방법에 의하면 출력 유닛 $y_k(t)(k=1,2,\cdots,n)$의 집합을 U, 입력 유닛 $x_k(t)(k=1,2,\cdots,m)$의 집합을 I라고 하면, 완전 연결형 리커런트 신경망은 출력과 입력을 하나로 결합한 $z_k(t)$에 의해 정의된다.

$$z_k(t)=\begin{cases} x_k(t) & \text{if } k\in I \\ y_k(t) & \text{if } k\in U \end{cases} \quad (2)$$

이때 k번째 출력에 들어오는 활성화 입력 $s_k(t)=\sum_{l\in U\cup I} w_{kl}(t)\cdot z_l(t)$이고, 출력 $y_k(t)=f_k(s_k(t))$이다. 여기서 f_k는 시그모이드 함수이다.

원하는 출력과 실제 출력 사이의 차를 $e_k(t)$라고 하면, j번째 입력 유닛과 i번째 출력 유닛사이의 가중치 $\Delta w_{ij}(t)$는 아래 식에 의해 갱신된다. 그러나, 주행 로봇 제어의 경우 원하는 출력과 실제 출력 사이의 차를 $e_k(t)$를 직접 구할 수 없으므로, 본 연구에서는 앞서의 CRBP 학습 알고리즘의 Step 5에서 얻어진 재강화 신호에 따른 확률적 오차를($E=S-O$ 또는 $E=O-S$) $e_k(t)$ 대신 사용한다.

$$\Delta w_{ij}(t)=\alpha\sum_{k\in U} e_k(t)\cdot p_{ij}^k(t) \quad (3)$$

여기서 α는 학습율이고, $p_{ij}^k(t)=\frac{\partial y_k(t)}{\partial w_{ij}(t)}$는 j번째 입력 유닛과 i번째 출력 유닛사이의 가중치 $w_{ij}(t)$의 변화에 대한 k번째 출력 $y_k(t)$의 변화율을 나타내는데 아래 식에 의해서 결정된다.

$$p_{ij}^k(t+1)=f'(s_k(t))\Big[\sum_{l\in U} w_{kl}\cdot p_{ij}^l(t)+\delta_{ik}\cdot z_j(t)\Big] \quad (4)$$

여기서 초기 조건은 $p_{ij}^k(0)=0$이고, $f'(s_k(t))$는 활성화 값 $s_k(t)$에 대한 출력의 미분치이고, δ_{ik}는 Kronecker 델터이고, $z_j(t)$는 $j\in U\cup I$에 속하는 유닛의 출력값이다.

위 식으로부터, k번째 출력 유닛은 j번째 입력 유닛과 i번째 출력 유닛 사이의 모든 연결 가중치의 $p_{ij}^k(t)$ $(i=1,2,\cdots,n,\ j=1,2,\cdots,m)$를 저장해야 하므로 일반적인 역전파 알고리즘에 비해 n배만큼 더 많은 덧셈과

곱셈이 요구된다. 이상의 학습 알고리즘은 단층 퍼셉트론 구조에 대해 설명하였는데, 은닉층을 갖는 다층 퍼셉트론의 경우도 위의 학습 알고리즘을 그대로 확장하여 사용하면 된다.

2. 유전 알고리즘에 의한 신경망 제어기 진화

앞 절에서 설명한 CRBP 학습에 의한 신경망 제어기의 가중치 결정은 얻어진 신경망 제어기의 성능이 초기 가중치 및 학습 파라메터에 많은 영향을 받을 뿐 아니라, 국부해 문제에 의해 항상 최적해를 보장하지 못하는 단점이 있다. 이를 극복하고자 최근 자연계의 적자 생존과 유전 현상을 모방한 선택과 재생 연산을 사용하는 해집단 기반 유전 알고리즘(GA: Genetic Algorithm)을 이용하여 주어진 문제에 대한 최적해를 진화에 의해 얻고자 하는 많은 연구가 수행되고 있다[15,16].

유전 알고리즘은 자연계의 적자 생존과 유전현상을 모방한 일종의 탐색 알고리즘으로 Michigan대학의 John Holland와 그의 동료 및 학생들에 의해 개발되었다[15]. 유전 알고리즘에서는 해결하고자 하는 문제를 일차원적 string형태로 표현하고, 여기에 유전 연산자를 적용하여 얻어지는 새로운 해가 점차로 최적해로 접근해 나간다. GA의 동작 과정은 다음과 같다. N 개의 해개체로 구성된 해집단을 P 라고 한다. 무작위로 생성된 초기 해집단을 $P(0)$ 라하고 시간 t 의 해집단을 $P(t)$ 라 한다. $P(t)$ 에 유전 연산(재생, 교차, 변이)을 적용하여 새로운 해집단 $P(t+1)$ 를 얻는다. $P(t+1)$ 에서의 각 해개체들은 시간 t 에서의 목적 함수값에 비례하여 재생된 것들이다. 교차는 두 해개체를 임의의 위치에서 절단하고, 첫 번째 해개체의 일부분이 다른 하나의 해개체로 옮기는 등의 방법으로 유전 물질을 서로 교환하여 재결합한다. 변이는 임의로 선택된 유전자의 값의 일부를 변경한다. GA의 전체적 효과는 더 높은 목적 함수값을 갖는 더 나은 해를 갖도록 새로운 해집단을 생성해 나가는 것이다. 그림 3은 GA의 동작 과정을 pseudo-code로 표현한 것이다.

```
Begin
    t = 0;
    Initialize P(t);
    Evaluate fitness functions in P(t);
    while ~(stop_conditions) do
    Begin
        t = t + 1;
        Select P(t) from P(t-1);
        Recombine chromosomes in P(t);
        Evaluate fitness functions in P(t);
    End
```

그림 3. 유전 알고리즘의 일반적인 동작 과정
Fig. 3. General procedure of genetic algorithms.

주행 로봇의 최적 주행 및 충돌 회피를 위한 신경망 제어기의 가중치를 결정하기 위해 GA를 사용할 때, 다음의 다섯 가지 문제를 고려하여야 한다.

- 문제에 적절한 해개체 표현 방법
- 초기 해집단의 생성 방법
- 제어 목적에 대한 우수성을 평가하는 적합도 함수
- 새로운 해집단을 생성하는 유전 연산
- GA에 제공되는 초기 파라메터들의 집합

(1) 해개체 표현

본 연구에서는 사용된 신경망 제어기는 입력 및 출력 유닛이 각각 8개 및 2개인 단층 완전 연결형 리커런트 신경망 구조를 가지므로 가중치 연결의 개수는 두 층간의 $(8+1)\times 2$개의 가중치와 출력층 유닛간의 리커런트 연결 가중치 4개를 합하면 전체 22개가 된다. 여기서 입력층의 유닛이 9개인 것은 바이어스를 포함했기 때문이다. 따라서, 한 신경망 제어기를 나타내는 해개체는 22개의 가중치에 대응하는 22개의 유전 인자값에 의해 표현되며, 각 유전 인자값은 실수값을 갖는다고 가정한다. 22개의 실수값 중 처음 18개는 두 층간 가중치이고 뒤따르는 4개의 실수값은 출력층의 리커런트 연결 가중치를 나타낸다. 해개체내 각 유전 인자들의 값이 실수이므로 해개체 표현은 실수 표현[17]을 사용한다.

(2) 초기 해집단 결정

초기 해집단의 결정은 GA의 수렴 속도나 얻어진 신경망 제어기의 주행 및 충돌 회피 능력을 결정하는데 많은 영향을 끼친다. 본 연구에서는 신경망 제어기의 초기 가중치를 $[-1, 1]$사이의 임의의 값을 난수 발생시켜 할당하였다.

(3) 목적 함수

본 연구에서 사용된 주행 로봇의 신경망 제어기가 갖는 제어 목적은 벽에 부딪히지 않고 오래 동안 주행하는 것으로 이를 위해 다음과 같이 세 개의 성분으로

구성된 목적 함수를 정의한다.

$$F = v(1-\sqrt{\Delta v})(1-i) \tag{5}$$

여기서, $v(0<v<1)$는 두 바퀴의 평균 회전 속도이고, $\Delta v(0<\Delta v<1)$는 전진은 양수, 후진은 음수로 나타내어지는 두 바퀴의 속도의 산술적 차의 절대값이고, $i(0<i<1)$는 8개의 입력 센서중 가장 활성화 값이 높은(즉 벽에서 가장 가까운) 센서가 갖는 활성화 값이다. 따라서, 목적 함수중 첫 번째 성분 v는 얼마나 주행 로봇이 빨리 움직이는가를 나타내고, 두 번째 성분 $1-\sqrt{\Delta v}$는 주행 로봇이 얼마나 직선으로 움직이는가를 나타내며, 세 번째 성분 $(1-i)$는 주행 로봇이 얼마나 충돌하지 않고 주행하는가를 나타낸다. 주행 로봇이 원형이고, 바퀴가 양방향으로 화전이 가능하므로, 목적 함수 F는 각 주행 방향에 하나씩 두 개의 동일한 최대값을 갖는 대칭적 표면을 갖는다.

(4) 유전 연산

주어진 외부 환경에서 벽에 부딪히지 않고 신속하게 주행을 하는 최적의 신경망 제어기를 얻기 위해 임의로 얻은 신경망 제어기의 초기 해집단에 아래에 설명한 유전 연산을 적절히 적용하여 진화시킨다.

1) 재생

해개체들을 재생하기 위해 여러 선택 방법들을 혼합하여 사용한다. 먼저 최고해 선택(elitism) 방식에 의해 목적 함수값이 가장 큰 해개체는 그대로 다음 세대에 전달된다. 두 번째 선택 방법은 k-토너먼트 방법[18]을 수정하여 사용하였다. 이 방법은 목적 함수값이 큰 해개체 들로부터 임의로 k개를 선택한 후, 이들 가운데 가장 큰 목적 함수값을 갖는 해개체를 선택한다. 위의 과정을 반복 적용하여 얻어 진 두 코드북 해개체 C'와 C''에 대해 다음에 설명할 교차와 변이를 적용하여 새로운 해개체 C를 생성한다. 목적 함수값이 작은 해개체 들로부터 임의로 k개를 선택한 후, 이들 가운데 가장 작은 목적 함수값을 갖는 해개체 $\breve{C}$는 새로운 해개체 C에 의해 교체된다. 위의 재생 과정을 $pselect\times|P|$ (여기서 $|P|$는 해집단의 크기)만큼 반복한다. 마지막으로, 해집단의 나머지 부분은 목적 함수값이 큰 순서로 복사하여 채운다. 그림 4는 위의 세 가지 재생 방법을 혼합한 hybrid 재생 방법을 보인 것이다.

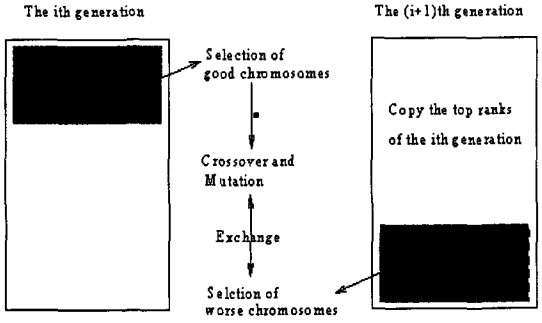

그림 4. 제안한 hybrid 재생 방법
Fig. 4. Proposed hybrid selection method.

2) 교차

위의 재생 과정에서 선택된 두 신경망 제어기에 대응하는 두 해개체 C'와 C''간의 교차는 다음과 같이 이루어진다. 먼저, 선택된 두 해개체가 갖는 22개의 유전 인자 위치 중 임의로 교차시키는 유전 인자를 교차 확률 P_c 만큼 선택한다. 다음, 두 해개체에서 선택된 유전 인자의 값을 각각 w'과 w''이라고 하면 새로운 해개체 C의 대응하는 위치에서의 유전 인자의 값 w는 다음과 같은 산술 평균에 의해 구해진다.

$$w = \frac{w' + w''}{2} \tag{6}$$

3) 변이

새로 얻어진 해개체 C에 대한 돌연 변이는 다음과 같이 이루어진다. 먼저, 얻어진 해개체 C의 22개의 유전 인자 위치 중 돌연 변이 시킬 유전 인자를 돌연 변이 확률 P_m만큼 선택한다. 다음, 해개체 C에서 선택된 유전 인자의 값을 w라고 하면 돌연 변이된 해개체 $\breve{C}$의 대응하는 위치에서의 유전 인자의 값 $\breve{w}$는 다음과 같은 반전에 의해 구해진다.

$$\breve{w} = -w + random[-\Delta w, \Delta w] \tag{7}$$

여기서, $random[-\Delta w, \Delta w]$는 $[-\Delta w, \Delta w]$사이에서 발생시킨 임의의 난수 값을 나타낸다.

Ⅲ. 진화와 학습의 동시 적응에 의한 신경망 제어기 설계

앞장에서 신경망 제어기 설계시 학습과 진화가 어떻게 이용되는지를 알아보았다. 본 장에서는 최근 생물학적 실험 결과에서 얻은 개념을 바탕으로 진화와 학습을 결합하는 방안을 설명하고자 한다. 앞에서 설명한 것처럼 학습이 진화에 어떻게 영향을 미치는 가에는

두 가지 가설: Lamarckian 가설과 Baldwin 효과가 있다. 한 생명체가 일생 동안 얻은 표현형의 특성이 유전될 수 있는 유전자형에 복사되어 한 생명체의 자손에 직접 전달된다는 Lamarckian 가설[8]은 오늘날 생물학적 견해에 따르면 더 이상 사실이 아니다. 그러나 컴퓨터 공학의 입장에서는 Lamarckian 가설이 컴퓨터 상에 쉽게 구현될 수 있고, 동시에 그것이 갖는 빠른 수렴 특성으로 인해 최적화 문제에 훨씬 더 효과적으로 이용될 수 있음을 알 수 있다. 따라서 본 연구에서는 Lamarckian 가설에 바탕을 둔 학습과 진화의 상호 적응 방법[19]을 주행 로봇을 위한 최적의 신경망 제어기를 설계 문제에 응용하고자 한다.

그림 5는 최적 신경망 제어기를 얻기 위한 Lamackian 상호 적응 과정을 나타낸 것이다. t번째 세대에서 i번째 해개체가 목적 함수값 $F_i(t)$를 갖는다고 가정한다. 각 해개체는 일생동안 앞장에서 설명한 CRBP 학습 알고리즘에 의해 학습된다. 각 해개체는 서로 다른 초기 파라메터의 설정에 따라 서로 다른 환경하에서 서로 다르게 학습되게 되는데 각 해개체의 학습 후 목적 함수값을 $F_i'(t)$라고 한다. 학습 후 어떤 해개체는 더 나은 목적 함수값을 가질 수 있고 ($F_i'(t) > F_i(t)$), 어떤 해개체는 더 못한 목적 함수값을 가질 수 있다 ($F_i'(t) < F_i(t)$). 본 연구에서는 학습된 신경망의 가중치 $\boldsymbol{w}'$를 다음과 같은 우성적(dominant) 해개체 선택에 의해 더 나은 목적 함수값을 갖는 경우의 가중치를 선택하였다.

$$\boldsymbol{w}^l = \begin{cases} \boldsymbol{w} & \text{if } F_i(t) > F_i'(t) \\ \boldsymbol{w}' & \text{if } F_i(t) < F_i'(t) \end{cases} \tag{8}$$

여기서, $\boldsymbol{w}$와 $\boldsymbol{w}'$는 각각 학습 전후의 신경망의 가중치를 의미한다. 위 해개체 선택법은 학습에 의해 가중치가 더 나쁜 목적 함수를 나타내면 비록 학습이 진행되었더라도 학습된 가중치가 선택되지 않음을 의미한다.

학습을 고려하지 않고 진화만 고려한다면, 가중치가 학습에 의해서 변경되는 과정이 생략되므로 학습된 가중치 $\boldsymbol{w}^l$과 학습전 가중치 $\boldsymbol{w}$는 서로 갖게 된다. 각 해개체가 학습된 다음 얻어진 가중치는 앞 장에서 설명한 유전 알고리즘에 의해 진화된 가중치 $\boldsymbol{w}^{l+e}$를 갖는 $(t+1)$번째 세대의 신경망 제어기 해집단이 얻어진다.

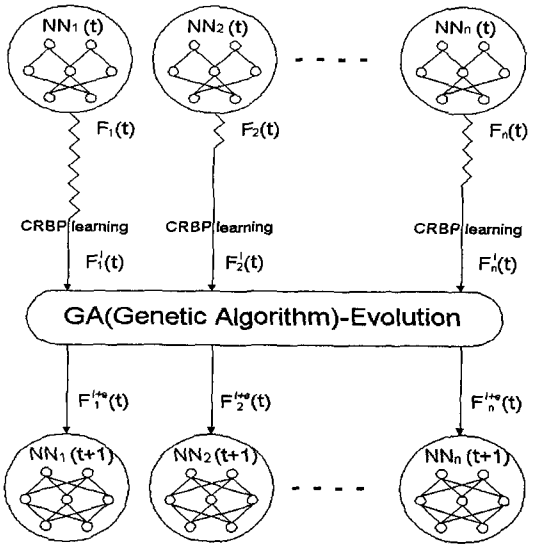

그림 5. 최적 신경망 제어기를 얻기 위한 Lamarckian 상호 적응

Fig. 5. Lamarckian co-adaptation for optimal NN controller.

이상의 설명으로부터 최적 신경망 제어기를 얻기 위한 Lamarckian 상호 적응 알고리즘은 다음과 같이 나타내어진다.

Step 1 : 초기화

학습 집단 $T = \{e_1, e_2, \ldots, e_{N_T}\}$를 준비하고 반복 시간 t를 $t=0$ 으로 한다.

주어진 수 N에 따라 N개의 해개체 $C(t) = \{C_1(t), C_2(t), \ldots, C_N(t)\}$를 만든다. 여기서, 각 해개체는 각 신경망 제어기에 대응한다. 각 신경망 제어기는 22개의 가중치의 나열에 의해 표현되는데 각 가중치의 초기값은 $[-1, 1]$사이에 한 값을 무작위로 발생시켜 얻는다.

Step 2 : 각 신경망의 학습전 적합도 $F_i(t)$ 계산(식 (5) 이용)

Step 3 : 각 신경망의 학습(training)

각 신경망의 가중치가 더 이상 변하지 않을 때까지 CRBP 학습 규칙에 따라 학습한다. 각 신경망의 학습 시간은 각 신경망이 갖는 초기 파라메터와 학습 환경에 따라 달라질 수 있다.

Step 4 : 각 신경망의 학습후 적합도 $F_i'(t)$ 계산 (식 (5) 이용)

Step 5 : 학습된 신경망의 가중치 $\boldsymbol{w}^l$를 식 (8)에 계산

Step 6 : 학습된 신경망 가중치에 대해 유전 알고리즘을 사용하여 진화

Step 5에서 계산한 적합도에 기반하여 제안한 hybrid 선택 방법을 사용한다.

선택된 해개체에 식 (6)과 (7)을 이용하여 교차와 변이를 적용한다.

Step 7 : 학습과 진화가 수행된 각 신경망 제어기의 가중치 $\boldsymbol{w}^{l+e}$를 계산

Step 8 : 학습과 진화가 수행된 각 신경망의 적합도 $F_i^{l+e}(t)$ 계산(식 (5) 이용)

해집단내 최대 목적 함수를 갖는 해개체를 $C_{best}(t)$라고 한다. 만약 $C_{best}(t)$의 주행 및 충돌 회피 능력이 원하는 수준 이상 최적 신경망 제어기를 $C_{best}(t)$라하고 알고리즘 수행을 중단한다.

Step 9 : $t=t+1$로 하고 Step 2로 간다.

그림 6. 제안한 최적 신경망 제어기 동시 적응 알고리즘
Fig. 6. Proposed co-adaptive algorithm for optimal NN controller.

Ⅳ. 실험 결과 및 분석

본 연구에서 사용된 주행 로봇은 Khepera 로봇[20]으로 직경 5.5 cm, 높이 5 cm, 무게 70g의 원형의 소형 로봇으로 8개의 적외선 센서가 전방에 6개, 후방에 2개가 원주 상에 놓여 있다. 각 센서는 0에서 1023까지의 값을 가지며 값이 클수록 물체에 근접함을 나타낸다. 로봇의 움직임은 2개의 DC 모터에 의해 구동된 원형 바퀴에 의해서 이루어지는데, 각 모터는 -10에서 10사이의 값을 가지며 음의 값은 후진을 의미하고 양의 값은 전진을 나타낸다. 그림 7은 사용된 Khepera 주행 로봇의 전경과 구성을 나타낸다.

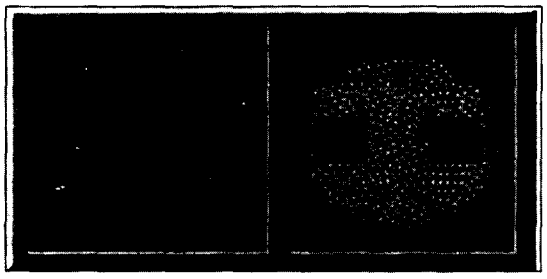

그림 7. Khepera 주행 로봇과 구성
Fig. 7. Khepera robot and its organization.

Nice Sophia 대학의 Olivier Michel[21]은 Khepera 주행 로봇을 Unix 워크스테이션 상에서 시뮬레이션 할 수 있는 Khepera 시뮬레이터를 X11 그래픽 환경 하에서 개발하였다. 이 시뮬레이터는 로봇을 구동하고 구동 결과를 표시하는 여러 함수 라이브러리를 제공하며, 사용자가 C 또는 C++ 언어를 사용하여 제어 알고리즘을 작성하여 테스트할 수 있도록 되어 있다. Khepera 시뮬레이터가 제공하는 함수에는 1) 좌우 모터 값을 제어하는 (-10에서 10 사이의 값 : short integer 형) robot->Motor[LEFT].Value와 robot->Motor[RIGHT].Value이 있고, 2) 적외선 센서의 값을 있는 (0에서 1023사이의 값 : short integer 형) robot->IRSensor[n].DistanceValue이 있고, 3) 로봇의 위치 및 각도를 나타내는 robot->X, robot->Y, robot->Alpha가 있고, 4) 시뮬레이터 프로그램 Fill-in 함수로 특정 버튼을 클릭했을 때 실행되는 void RunRobotStart(struct Robot *robot), boolean StepRobot (struct Robot *robot), void RunRobotStop(struct Robot *robot), void UserInit(struct Robot *robot), void UserClose(struct Robot *robot)등이 있다. Khepera 시뮬레이터는 로봇의 환경하에서 로봇의 주행 상태를 관측할 수 있는 'world' 부분과 로봇 내에서 일어나는 센서값, 구동 모터 크기, 및 제어기의 제어 값등을 볼 수 있는 'robot' 부분 두 가지로 나뉘어져 있다. 그림 8은 본 연구에서 사용된 복도 모양의 외부 world를 갖는 Khepera 시뮬레이터의 초기 화면을 나타낸 것이다.

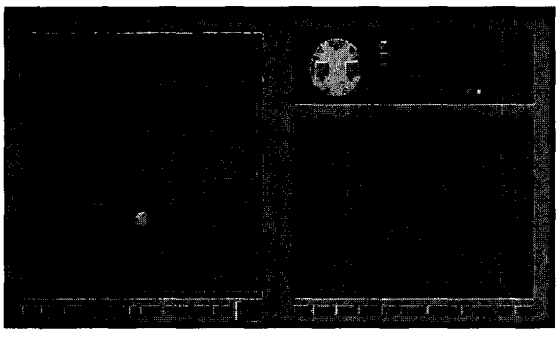

그림 8. Khepera 시뮬레이터 초기 화면
Fig. 8. Initial display of Khepera simulator.

주어진 환경에서 로봇이 주행하는 과정을 실험하는 방안은 크게 두 가지가 있다. 하나는 주어진 외부 환경

하에서 실제 물리적 로봇을 움직여서 관측하는 것이고, 다른 하나는 로봇 시뮬레이터를 이용하여 가상의 외부 환경에서 수치적 로봇을 움직여서 관측하는 것이다. 두 방법은 각각 장단점을 갖고 있다. 일반적으로 시뮬레이터를 이용한 관측은 빠르고 값싸게 실험을 할 수 있으며, 실험 수행 중 나타나는 여러 변수들의 완전한 제어와 기록이 가능하다는 장점이 있다. 그러나 수치적 시뮬레이션은 로봇과 로봇이 처한 외부 환경 사이의 상호 작용의 모든 물리적 법칙(예를 들면, 무게, 질량, 마찰, 중력등), 오동작, 부품 불량, 피로 등과 같은 요인을 모두 고려하기가 어려운 단점이 있다[22]. 따라서 본 연구에서는 위의 두 방안의 혼합적인 방법을 사용하고자 한다. 즉, CRBP학습과 유전 알고리즘과 같이 연산 시간이 많이 걸리는 부분은 컴퓨터상에서 시뮬레이터상에서 수행하고, 연산 결과 얻어진 구동 데이터는 컴퓨터에 연결된 선을 따라 물리적 로봇에 보내어 실제 외부 환경 하에서 로봇의 주행을 관측한 후 관측 결과를 다시 연결된 선을 따라 컴퓨터에 보내어 이를 컴퓨터상에서 분석하는 방법을 사용하였다. 그림 9는 본 실험에서 Lamarckian 상호 적응시 사용된 학습과 진화 파라메터를 나타낸 것으로 α와 β는 각각 상벌(reward and punishment)시 학습율을 나타내며, 학습 주기당 스텝 수는 학습시 한 시작점에서 주행 로봇을 구동한 횟수를 나타내며, 초기 시작점 수는 몇 개의 시작점에서 학습 과정을 수행했는가를 나타낸다.

해집단 크기	100
세대수	100
해개체 크기	22
Pc	0.5
Pm	0.2
α	0.3
β	0.1
학습주기당 스텝수	1000
초기 시작점 수	4

그림 9. Lamarckian 상호 적응시 수행 파라메터
Fig. 9. Execution parameters for Lamarckian co-adaptation.

그림 10은 본 연구에서 고려한 자율적 이동 로봇의 주행 및 충돌 회피 실험을 위해 사용된Khepera 로봇과 제안한 진화와 학습의 Lamarckian 상호 작용 실험을 위해 사용된 복도 모양의 환경을 나타낸 것이다. Khepera 로봇을 제어하는 신경망 제어기는 Linux 환경하에서 동작하는 PC상에서 얻어지고 얻어진 제어기의 제어 프로그램이 직렬 연결선을 통해 Khepera 로봇을 통해 download 되어 움직이게 되는데 실제 환경 하에서 동작하는 신경망 제어기의 주행 및 회피 능력은 임의의 주행 각도를 갖는 100개의 시작점에서 매 시작점마다 1,000 스텝 Khepera 로봇을 주행시킬때 식 (5)에 의해 얻어지는 Fitness값에 의해 평가된다.

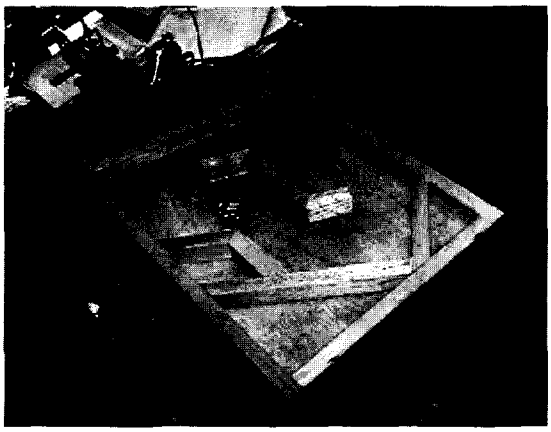

그림 10. Khepera 로봇과 외부 주행 환경
Fig. 10. Khepera robot and its traveling environment.

그림 11은 주어진 외부 환경 하에서 제안한 진화와 학습의 Lamarckian 상호 적응에 신경망 제어기를 상호 적응시켰을 때 각 세대 당 평균 목적 함수값의 변화 곡선을 전체 목적 함수값과 목적 함수의 세 구성 성분 v, $1-\sqrt{\Delta v}$, 그리고 $(1-i)$별로 나타낸 것이다. 이 그림으로부터 로봇의 주행 속도를 나타내는 목적 함수 v는 진화가 진행되면서 계속적으로 증가하는 경향을 보이고 있으며, 로봇의 직진성을 나타내는 목적 함수 $1-\sqrt{\Delta v}$는 진화 초기에 최대치에 가까운 값에 도달한 후 큰 변화가 없으며, 로봇의 충돌 회피 능력을 나타내는 목적 함수 $(1-i)$는 진화가 진행되면서 완만하게 증가하지만 증가율은 그다지 크지 못함을 알 수 있고, 이들의 곱에 의해 정의된 전체 목적 함수 $F=v\cdot(1-\sqrt{\Delta v})\cdot(1-i)$는 진화가 진행되면서 증가하는 경향이 뚜렷이 나타남을 알 수 있다. 따라서, GA에 의한 진화시 목적 함수의 증가에 가장 많은 영향을 미

치는 요인은 로봇의 주행 속도임을 알 수 있는데 이는 로봇의 주행을 관측할 때 진화 초기에는 로봇이 거의 움직이지 못하고 제자리에 머무르면서 움직이지 못하지만 진화가 진행되면서 로봇이 점차 빠르게 움직이는 결과와 서로 일치한다.

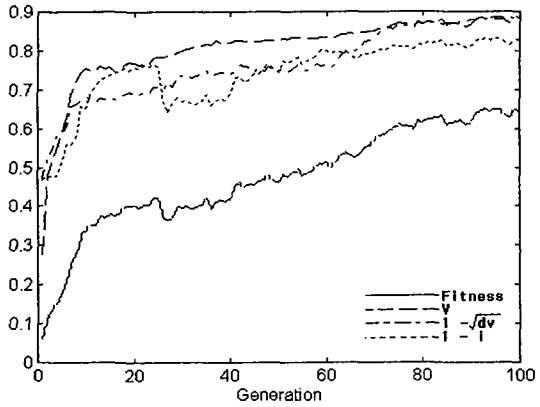

그림 11. Lamarckian 상호 적응에 의한 목적 함수의 진화 곡선

Fig. 11. Evolution curve of fitness function in Lamarckian co-adaptation.

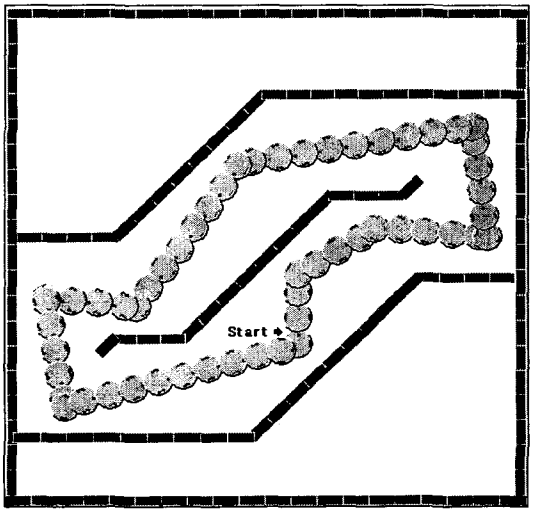

그림 12. 임의의 한 점에서 시작한 Khepera 로봇의 주행 경로

Fig. 12. A tracing trajectory of Khepera robot starting a random initial point.

얻어진 최적의 신경망 제어기의 주행 및 충돌 회피 능력을 테스트하기 위해 여러 시작점에 대해 측정하기 위해, 임의의 주행 각도를 갖는 100개의 시작점에서 Khepera 로봇을 주행시키는 실험을 하였는데, 이 실험 결과 100개의 임의의 점에 대해 충돌이 일어나는 경우는 한 번도 일어나지 않고 초당 약 5cm의 매우 빠른 속도로 움직이는 것이 관측되었다. 그림 12는 임의의 한 시작점에 대한 계속 반복되는 주행 경로의 한 주기를 나타낸 것이다.

Ⅳ. 결 론

본 논문은 Lamarckian 동시 적응과 같은 생물학적으로 얻은 개념을 이용한 주행 로봇의 신경망 제어기의 설계 방법을 제안하였다. CRBP와 같은 학습 알고리즘에 의해 신경망 설계는 수렴 결과 얻어진 제어기의 주행 및 충돌 회피 능력이 초기 가중치와 학습 파라메터에 매우 민감하다는 단점을 가지고 있다. 특히, 신경망 제어기의 입출력 구조가 복잡하여 결정해야 하는 가중치가 많은 경우 주어진 문제의 최적해를 보장해주는 초기 가중치와 초기 학습 파라메터를 결정하는 것은 거의 불가능하다. 또, 제안한 학습 알고리즘이 확률론적 경사 강하 방법(gradient descent method)을 사용하고 있으므로 수렴 여부를 보장하기가 어려운 단점이 있다.

이러한 단점들을 극복하기 위해, 유전 알고리즘에 기반하여 신경망 제어기를 진화시켜 주어진 제어 목적을 잘 만족하는 방안을 제안하였다. 유전 알고리즘은 그것이 갖는 내부적 병렬성(implicit parallelism)으로 인해 문제의 해공간을 전역적으로 탐색한다. 이 방법의 한가지 단점은 유전 알고리즘이 해집단을 기반으로 동작하기 때문에 최적해에 수렴하는데 까지 진화시간이 오래 걸리고, 주어진 해 근처의 국부적 탐색이 어려운 단점이 있다. 이를 해결하기 위해 본 연구에서는 생물학적 견해의 하나인 한 생명체가 일생 동안 얻은 표현형(phenotypic)의 특성이 유전될 수 있는 유전자(genetic)형에 복사되어 한 생명체의 자손에 직접 전달된다는 Lamarckian 가설을 이용하여 학습과 진화의 교대적 적용에 의한 신경망 제어기의 설계 방안을 제안하였다. 이 설계 방법은 GA에 의해 해집단내 여러 개의 신경망 제어기들은 전역적 탐색을 수행하여 점진적으로 장애물과의 충돌이 적게 일어나도록 진화되고, 동시에 각 신경망 제어기는 상보적 재강화 역전파 (CRBP : Complementary Reinforcement Backpropagation) 학습에 의해 국부적 탐색을 수행하여 주행 특성이 로봇이

처한 외부 환경에 적응되어지도록 한다.

이와 같은 Lamarckian 동시 적응 방법을 신경망 제어기 설계 문제에 효과적으로 적용한 동시적응 연산 모델로서 CRBP 학습과 유전 알고리즘과 같이 연산 시간이 많이 걸리는 부분은 Khepera 로봇 시뮬레이터상에서 수행하고, 연산 결과 얻어진 구동 데이터는 컴퓨터에 연결된 선을 따라 실제 물리적 로봇에 보내어 실제 외부 환경 하에서 로봇의 주행을 관측한 후 관측 결과를 다시 연결된 선을 따라 컴퓨터에 보내어 이를 컴퓨터 상에서 분석하는 방법을 사용하였다. 실험 결과, 학습과 결합한 진화에 의해 얻어진 신경망 제어기가 진화 자체만에 의해 얻어진 신경망 제어기보다 더 나은 충돌 회피 능력을 보여 주며, 원하는 주행 성능에 보다 빨리 도달하는 것을 확인할 수 있다. 결론적으로, 진화와 학습의 Lamarckian 상호 적응에 의한 주행 로봇의 설계 방안은 CRBP 학습이나 GA 진화만으로 수행하는 경우보다 최적 신경망 제어기를 얻을 수 있는 가능성이 더욱 증가하고, 보다 더 빨리 수렴한다는 점에서 주행 로봇을 위한 최적의 신경망 제어기 설계 문제에 매우 효과적임을 알 수 있다.

참 고 문 헌

[1] L. Steels, "Building agents out of autonomous behavior systems," *Building Simulated embodied Agents*, New Haven: Lowrence Erlbaum, 1993.

[2] L. Meeden, "An incremental approach to developing intelligent nerual network controllers for robots," *IEEE Transactions on Systems, Man, and Cybernetics*," vol. 26, no. 3, pp. 474-485, 1996.

[3] I. Harvey, P. Husbands, and D. Cliff, "Issues in evolutionary robotics," *From Animals to Animats* 2, J.-A. Meyer, H. Roitblat, and S. Wilson, Eds. Cambridge, MA: MITPress, 1993, pp. 364-373.

[4] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *IEEE Transaction on Systems, Man, and Cybernetics*," vol. 26, no. 3, pp. 396-407, 1996.

[5] J. Millán, "Rapid, safe, and incremental learning of navigation stragegies," *IEEE Transaction on Systems, Man, and Cybernetics*," vol. 26, no. 3, pp. 408-420, 1996.

[6] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks : Using the genetic algorithm with connectionist learning," *Artificial Life II*, edited by C. G. Langton, J. D. Farmer, S. Rasmussen, and C. E. Taylor, pp. 511-548, Addison-Wesley, 1991.

[7] G. F. Miller, P. M. Todd, and S. U. Hedge, "Designing neural networks using genetic algorithms," *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[8] J. B. Lamarck, "Of the influence of the environment on the activities and habits of animals," *it Zoological Philosophy*, Macmillan, London, 1914.

[9] D. H. Ackley, and M. L. Littman, "A case for Lamarckian evolution," *Artificial Life III*, edited by C. G. Langton, pp. 3-10, Addison-Wesley, 1994.

[10] J. M. Baldwin, "A new factor in evolution," *The American Naturalist*, vol 30, June, 1896.

[11] G. E. Hinton and S. J. Nowlan, "How learning can guide evolution," *Adaptive Individuals in Evolving Populations : Models and Algorithms*, edited by R. K. Belew and M. Mitchell, pp. 447-454, Addison Wesley, 1996.

[12] D. Parisi, S. Nolfi, "The influence of learning on evolution," *Adaptive Individuals in Evolving Populations : Models and Algorithms*, edited by R. K. Belew and M. Mitchell, pp. 419-430, Addison Wesley, 1996.

[13] D. H. Ackley, and M. L. Littman, "Generalization and scaling in reinforcement learning," in Advances in Neural Information Processing System 2, D. S. Touretsky, Ed. San Mateo, CA: Morgan Kaufmann, 1990. pp. 550-557.

[14] R. Williams, and David Zipser, "A learning

algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, pp. 270-280, 1989.

[15] J. H. Holland, *Adaptation in natural and artificial systems*, University of Michigan Press, 1975.

[16] David E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley Press, 1989.

[17] A. H. Wright, "Genetic algorithms for real parameter optimization," *Foundations of Genetic Algorithms*, edited by G. Rawlins, pp. 205-220, Morgan Kaufmann Publishers, 1991.

[18] Daijin Kim and Chulhyun Kim, "Forecasting Time Series with Genetic Fuzzy Predictor Ensemble", *IEEE Trans. on Fuzzy Systems*, vol. 5, no. 4, 1997.

[19] Daijin Kim, Sunha Ahn and Dae-Seong Kang, "Co-adaptation of Self-Organizing Maps by Evolution and Learning," *NeuroComputing*, vol. 30, pp. 249-272, 2000.

[20] F. Mondada, E. Pranzi, and P. Ienne, "Mobil robot miniaturization: A tool for investigation in control algorithms," Proceedings of the Thrid International Symopsium on Experimental Robotics, Kyoto, Japan, 1993.

[21] O. Michel, "Khepera Simulator v2.0", User's Manual, University of Nice, 1996.

[22] F. Mondada, and D. Floreano, "Evolution of neural control structures: some experiments on mobile robots," *Robotics and Autonomous Systems*, vol 16, pp. 183-195, 1995.

저 자 소 개

金 大 鎭(正會員) 第 36卷 S編 第 8號 參照
현재 포항공과대학교 컴퓨터공학과 부교수

姜 大 星(正會員) 第 36卷 S編 第 8號 參照
현재 동아대학교 전기전자컴퓨터공학부 조교수