유전 프로그래밍을 이용한 미지의 환경에서 상호 협력하는 로봇 제어기의 설계

論 文
48A - 9 - 14

Controller Design for Cooperative Robots in Unknown Environments using a Genetic Programming

丁 一 權* · 李 柱 張**
(Il-Kwon Jeong · Ju-Jang Lee)

Abstract - A rule based controller is constructed for multiple robots accomplishing a given task in unknown environments by using genetic programming. The example task is playing a simplified soccer game, and the controller for robots that governs emergent cooperative behavior is successfully found using the proposed procedure. A neural network controller constructed using the rule based controller is shown to be applicable in a more complex environment.

Key Words : 유전 프로그래밍, 협동 로봇, 로봇 축구, 신경회로망

1. 서 론

1970년대에 개발돼 최근 각광받고 있는 진화 연산 (Evolutionary Computation; EC) 이란 컴퓨터를 이용하여 적자생존을 토대로 생물학적 진화를 모의 실험하여 탐색문제의 해를 찾는데 이용하는 기법이다.

D. B. Fogel에 의하면 진화 연산은 크게 유전 알고리즘 (Genetic Algorithm; GA) 과 진화 알고리즘 (Evolutionary Algorithm; EA) 으로 나눌 수 있다[1]. 진화 알고리즘은 다시 진화 기법 (Evolution Strategy; ES) 과 진화 프로그래밍 (Evolutionary Programming; EP) 으로 나뉘어 진다. 이처럼 유전 알고리즘, 진화 기법 그리고 진화 프로그래밍이 진화 연산의 세 줄기를 형성하고 있으며 해를 표현하는 방법, 구체적인 해 변화 방법, 해의 선택방법에 따라 약간씩 차이가 있다. 이들 방법은 공통적으로 다음과 같은 동작을 하게 된다. 후보 해에 해당하는 각 개체들의 집단이 임의로 초기화된 후 선택, 돌연변이, 재결합과 같은 확률적 유전 연산자를 적절히 거쳐 탐색 공간상의 더 나은 공간으로 진화한다. 유전 알고리즘은 조합 최적화 문제에, 진화 프로그래밍 기법은 연속 최적화 문제에 잘 동작하는 특성이 있다. 제어 분야에선 특히 유전 알고리즘이 주로 적용되어 왔다. 대표적인 예로 시스템 동정화[2], 신경회로망 구조 그리고(또는) 가중치의 최적화[3][4] 그리고, 퍼지 제어기의 rule base 그리고(또는) 소속함수의 최적화[5][6]를 들 수 있다.

한편, 유전 프로그래밍 (Genetic Programming; GP) 이란 방법이 최근 J. R. Koza 에 의해 소개되었다[7]. 이는 컴퓨터 프로그램을 유전 알고리즘을 이용하여 진화시키는 방법으로서 유전 알고리즘의 확장된 형태라고 볼 수 있으며 부호화된 해 대신 함수와 터미널들로 이루어진 tree 구조를 사용한다. 유전 알고리즘은 풀고자 하는 문제에 따라 개체의 부호화 과정을 새로 정의해 주어야 하는 반면에 유전 프로그래밍을 사용하면 부호화 과정을 아주 간단하게 할 수 있게 되고 문제에 따라선 필요 없을 수도 있다. 다만, 사용되는 해의 구조를 정의해 주어야 한다. 유전 프로그래밍은 다음의 세 과정으로 이루어진다.

(1) 함수와 터미널들의 무작위 조합으로 이루어진 초기 해 (컴퓨터 프로그램)를 생성한다.
(2) 종료 조건이 만족될 때까지 다음의 과정을 반복한다.
  (a) 집단내의 각 개체를 실행시켜 얼마나 문제를 잘 풀었나에 따라 적합도를 부여한다.
  (b) 재생산, 교차변이 그리고 돌연변이와 같은 유전학적 연산자를 유전 알고리즘과 유사하게 적용하여 새로운 컴퓨터 프로그램들의 집단을 구성한다.
(3) 알고리즘이 종료될 때까지 생성되었던 프로그램 중에서 가장 성능이 뛰어난 개체가 유전 프로그래밍의 결과가 된다.

유전 프로그래밍에 관한 유용한 자료들은 Internet상의 web site를 통해서 많이 접할 수 있으며[10] 초기엔 LISP 언어로 구현되었으나 최근엔 속도 및 응용성이 향상된 C 또는 C++ 언어로 구현된 프로그램들이 많다. 본 연구에서는 구현의 용이성을 위하여 Bruno Haible이 만든 Common LISP 언어를 이용하여 따로 구현하였다.

상호 협력하는 로봇들의 제어는 로봇간의 상호 영향을 고려해야 하기에 복잡하게 된다. 미지의 환경일 경우엔 더욱 어려울 것이다. 하지만 이와 같은 상황에서 제어기의 설계에 시스템에 대한 정보뿐만 아니라 제어기 구조에 대한 정보도 필요로 하지 않는 방법이 있다면 아주 매력적일 것이다.

본 논문에서는 유전 프로그래밍을 사용하여 미지의 환경에서 제어기에 대한 구체적인 정보 없이, 여러 로봇이 상호

* 正 會 員 : 韓國電子通信研究院 先任研究員 · 工博
** 正 會 員 : 韓國科學技術院 電氣및電子工學科 敎授 · 工博
接受日字 : 1999年 1月 12日
最終完了 : 1999年 8月 6日

협력하여 하나의 목적을 달성하는 다개체 시스템에 대한 제어기를 설계하고자 한다. 유전 프로그래밍을 사용하여 얻어진 규칙 기반 제어기를 사용하여 신경 회로망 제어기를 구성할 수 있는지에 대해서도 분석하고자 한다.

2. 유전 프로그래밍을 이용한 제어기 설계 방법

2.1 배경

미지의 환경에서 상호 협력하도록 하는 로봇 제어기의 설계가 목표인 만큼 로봇 각각에 대한 복잡한 제어보다는 상호 작용과 행동 전략에 초점을 둔다. 각각의 로봇은 이동 로봇으로서 주위의 필요한 정보를 얻기 위한 센서가 부착된 것으로 본다. 여러 로봇이 하나의 목적을 달성하는 것이 목표이므로 각 로봇은 상호간의 작용 및 자신의 이동에 대한 행동 규칙을 가지고 있어야 한다. 같은 목적을 달성하는데 있어서 될 수 있으면 간단한 행동 규칙을 얻는 것을 목표로 하며 모든 로봇은 같은 행동 규칙을 가지고 공동의 목적을 달성할 수 있도록 한다.

유전 프로그래밍을 사용하여 얻게 되는 제어기는 if-then 규칙으로 이루어진 규칙 기반 제어기이지만, 이를 이용한 신경 회로망 제어기의 효과적인 구성 방법에 대해서도 논한다. 제안된 설계 방법의 타당성 검토를 위해 모의 실험을 하게 되며, 적용례로는 간략화된 이동 로봇 축구가 사용되었다.

본 연구와 유사한, 진화 연산을 이용한 다중 로봇의 제어에 대한 논문은 다음 세 편이 있다. Patel과 Maniezzo는 단순화된 축구 경기를 하는 로봇의 제어기로 신경회로망을 사용하고 그 신경망을 유전 알고리즘을 사용하여 학습시켰다[11]. 하지만, 결과가 별로 좋지 못했고 2명의 선수로 이루어진 한 팀만을 모의 실험하는 간단한 문제를 푸는데 그쳤으며, 유전 알고리즘을 이용한 신경망의 학습은 문제가 복잡해질수록 성능이 좋지 않으며 계산량이 늘어나는 단점이 있는 것으로 지적되고 있다[16][17]. Lohn과 Reggia는 자기 복제를 유도하는 automata 규칙을 유전 알고리즘을 사용하여 찾았다[12]. 그들은 effector automata(EA)라 이름 붙여진 변형된 cellular automata(CA) 모델을 사용하였다. CA 모델이 주위 환경에 반응하여 개체의 상태만이 변화하는데 반하여, EA 모델에서 각 개체는 cellular 공간에서 고유의 규칙 표를 사용하여 외부로부터의 입력에 대하여 해당하는 출력(이동, 복제, 소멸 등)을 내보내게 된다. 이 논문에서는 각 개체들의 상호 협력보다는, 단순한 자기 복제 능력에 초점을 맞추었으나, 축구하는 로봇과 같은 문제처럼 상호 작용하는 개체들의 행동이 요구될 때 EA 모델은 유용하게 적용될 수 있다. 저자는 역시 간략화된 축구 로봇 문제에 대하여 유전 알고리즘을 적용시킨바 있다[18]. 하지만, 3,300개의 규칙을 모두 포함하는 해를 표현하여야 하므로 계산량이 많아지고 메모리가 낭비되는 단점이 있었으며, 해의 크기가 변할 수 있는 유전 프로그래밍이 더 타당한 기법일 것이다.

2.2 함수와 터미널 집합

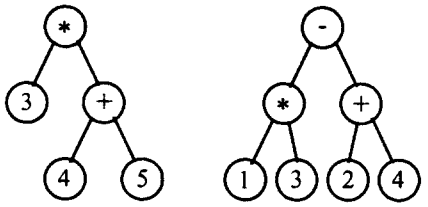
유전 프로그래밍은 유전 알고리즘과는 달리 해의 표현에 함수와 터미널들로 이루어진 구조를 사용한다. 유전 프로그래밍에 사용될 수 있는 함수들은 다음과 같이 분류할 수 있다. 산술 연산자 (+, -, ×, ÷ 등), 수학 함수 (sin, cos, exp, log 등), 논리 연산자 (and, or, not 등), 조건 연산자 (if-then-else 등), 반복 유도 함수(do-until 등), 재귀 함수 등이다. 터미널 집합에는 주로 대상 시스템의 상태 변수 등이 포함된다.

위에 열거한 모든 원소들로 이루어진 구조를 사용하여 해를 표현할 수 있겠지만 필요 이상으로 알고리즘이 복잡해질 우려가 있다. 따라서, 주어진 문제에 따라 어떤 함수를 선택할 것인가 하는 문제와 터미널 집합을 어떻게 구할 것인가 하는 문제가 생긴다. 터미널 집합은 보통 상태변수를 그 원소로 하는데 뒤에서 예로 들게 될, 간단한 축구 경기를 위한 이동 로봇의 경우엔 로봇의 행동 양식이 터미널이 된다.

2.3 유전 프로그래밍을 이용한 규칙 기반 제어기의 설계

유전 프로그래밍의 해는 LISP의 list 즉, S-expression으로 나타내는 것이 가장 자연스럽다. 그림 1에 tree 구조로 나타내어진 해의 예를, 그림 2에 그 해들의 S-expression을 나타내었다.

Tree 구조로 나타내어진 해는 노드(node)를 교차변이점으로 하여 교차변이를 하면 유효한 해가 구성되므로, 교차변이는 이를 만족하도록 구현하였고 (그림 3), 돌연변이는 논리 연산자나 터미널의 값을 임의로 바꿔주는 것으로 정의하였다.



3 * (4 + 5)　　(1 * 3) - (2 + 4)

그림 1 Tree 구조로 표시된 해의 예
Fig. 1 Solutions in the tree structure

(* 3 (+ 4 5))　　(- (* 1 3) (+ 2 4))

그림 2 그림 1의 tree 구조를 나타낸 S-expression
Fig. 2 S-expressions of the tree structures in Fig. 1

2.4 신경 회로망 제어기의 설계

본 논문에서는 신경 회로망을 직접 진화 연산 기법으로 학습시킬 때의 계산량 증가와 같은 단점을 피하기 위해 유전 프로그래밍으로부터 얻은 규칙 기반 제어기를 통해 신경 회로망 제어기를 구성하는 방법을 제안하고자 한다. 이를 위해서는 먼저 사용될 신경 회로망의 형태를 결정짓고 규칙 기반 제어기로부터 얻은 정보를 신경망에 학습시켜야 한다. 그러기 위해서는 먼저 얻어진 규칙들을 분석하여 학습에 사용될 규칙들을 정하고, 어떤 형태로 그 규칙들을 이용할 것

인지를 정하여야 한다. 즉, 학습 데이터로 바로 쓸 것인지 아니면 신경망의 특성을 고려하여 적절한 데이터의 변형을 가한 후에 적용할 것인지 정해야 한다. 아울러, 신경망에서는 어떠한 입력을 사용할 것인지 정해야 한다. 입력의 개수 및 출력의 개수 그리고 형태를 정해야 한다. 마지막으로, 어떠한 학습 방법을 사용할 것인지 결정하여야 한다. 다음 절에서 보다 구체적으로 설명하게 될 것이다.

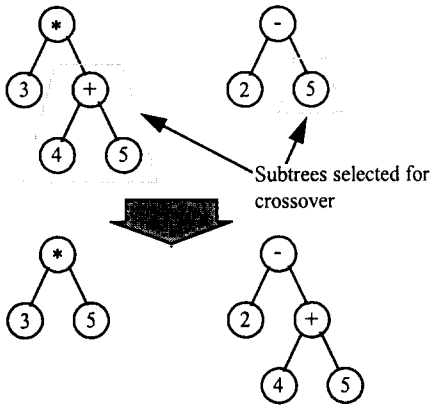


그림 3 교차변이의 예
Fig. 3 An example of crossover

3. 제안된 방법의 성능 평가 및 응용

구현된 유전 프로그래밍을 이용하여 이동 로봇이 축구 경기를 하는 문제에 적용을 해보았다. 이동 로봇 축구는 한국과학기술원에서 로봇 축구 대회(MIROSOT)가 개최된 후 지능 제어 및 비전 시스템 연구자 및 학생들의 관심을 끌고 있는 새로운 과학 기술 경연 도구로 등장했다. 여러 대의 로봇이 어울려 공동의 목표인 득점을 위해 협조해야 하므로, 본 연구에 적합한 응용 문제라 할 수 있을 것이다.

3.1 단순화된 축구 모델 (Simplified Soccer model; SS model)

모의실험을 위해 다개체 로봇 시스템에 적용이 용이한 EA 모델과 유사한 단순화된 축구 모델(SS 모델)을 제안하였다. SS 모델에서 개체는 로봇이 되고, 각 개체는 공통의 규칙표를 가지고 있으며 이 규칙표에 따라 주위의 환경을 입력으로 받아 출력을 내보내게 된다. 여기에서 출력은 '옆 칸으로 이동'과 같은 형태이다. 또, 이산화된 시간을 사용하며 2차원의 직사각형 모양을 이루는 여러 방(cell)들로 이루어져 있다. EA 모델에서 각 방에 개체 하나 만이 존재할 수 있는데 비해, SS 모델에서는 여러 개의 개체(로봇) 또는 공이 같은 방안에 존재할 수 있다고 가정하였다.

그림 4에 4대의 로봇으로 이루어진 SS 모델을 표시하였다. 각 로봇은 A1, A2, B1, B2로 표시되어 있으며(A와 B는 팀을 구별하는 문자이다) ●은 공을 나타낸다. 문제의 복잡도를 줄이기 위해 각 로봇은 상, 하, 좌, 우의 네 방향으로만 움직이거나 물체를 감지할 수 있다고 가정한다. 또, 각 로봇은 공이 어느 sector에 있는지 항상 감지할 수 있는 반면에 상대 선수(로봇)는 감지 창(view window)안에 있을 때만 포착할 수 있다(그림 5). 구체적으론 다음과 같은 규칙을 따른다. 같은 팀의 로봇일 경우엔 영역 I, II, III에서 감지된다. 상대방 로봇의 경우 영역 I에 존재하는 경우만 감지된다. 영역 III에 있는 골포스트는 감지할 수 있다. 공은 항상 감지될 수 있으며 공이 속하는 영역에 따라, 가운데, 상, 하, 좌, 우 중 하나의 형태로 감지된다(그림 5).

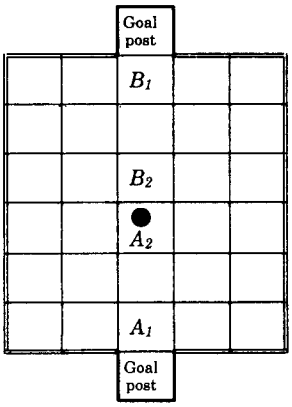


그림 4 SS 모델 (●는 공을 나타낸다.)
Fig. 4 SS model (● represents the ball)

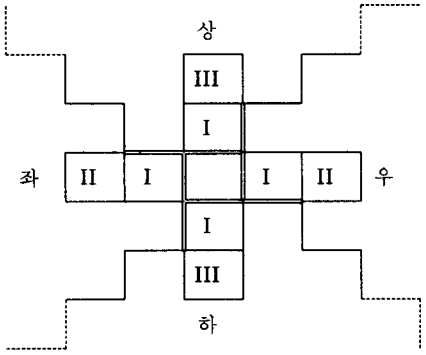


그림 5 로봇의 감지 창. 감지 창의 한 가운데는 로봇의 중심과 일치한다.
Fig. 5 View window of a robot. The center of the window corresponds to the center of a robot.

3.2 로봇의 행동 양식과 해의 표현 방법

각 로봇은 공통의 규칙표에 의해 제어된다. 규칙표는 규칙의 나열로 이루어져 있으며 각각의 규칙은 '조건-출력' 의 형태로 이뤄진다. 로봇의 출력에는 표 1에 나타낸 바와 같이 이동(move), 드리블(dribble), 킥(kick) 과 같이 3가지가 있다. 드리블과 킥은 로봇이 공을 가졌을 때만 수행할 수 있도록 제한하였다.

표 1에서 방향을 나타내는 [DIR]은 상, 하, 좌, 우 중의 한 값을 가질 수 있다. 이동과 드리블 시에 로봇은 한칸씩 이동할 수 있고, 킥을 할 때는 공만 두칸 보낼 수 있다. 로봇의 방향은 경기중 변하지 않는다고 가정함으로써 간접적으로 로봇들이 골포스트의 방향을 항상 알 수 있도록 하였다.

표 1 SS 모델에 사용되는 출력
Table 1 Actions used in the SS model

출 력	설 명
MOVE [DIR]	지정된 방향으로 한 간 이동한다.
DRIBBLE [DIR]	공을 가지고 있을 때에 한하여 지정된 방향으로 한 칸 드리블한다.
KICK [DIR]	공을 가지고 있을 때에 한하여 지정된 방향으로 공을 두 칸 찬다.

제안된 SS 모델에서는 구체적인 제어가 포함되어 있지 않으므로 유전 프로그래밍에 사용되는 함수에 산술 연산자는 포함되지 않았다. 로봇의 움직임을 직접 제어하는 제어기는 따로 구현가능함을 감안할 때, 본 논문의 규칙표에서의 규칙들은 상위 레벨의 행동 양식을 나타내는 행동 전략으로 생각할 수 있다. 함수는 논리 연산자와 조건 연산자로 이루어져 있다. 터미널 집합은 C, L, R, T, B (각각 중심, 좌, 우, 상, 하를 나타낸다)의 5개의 변수 및 출력으로 이루어져 있다. 각 변수는 로봇이나 공을 나타내는 기호(예: A1, B2, Ball)를 값으로 갖으며, 각 변수에 해당하는 위치에서 감지되는 물체를 나타낸다. 결국 해는 다음과 같은 S-expression이 된다.

$$(\text{progn}\ (\text{if}\ (\text{and}\ (\text{eq C A1})\ (\text{eq L Ball}))\ \text{MOVE}-\text{LEFT})\cdots) \quad (1)$$

모의 실험을 위해서 SS 모델의 시뮬레이터가 필요하다. 시뮬레이터는 규칙표에 따른 로봇과 공의 위치변화를 매 시간스텝마다 계산하고 득점을 기록한다. 시뮬레이터는 어느 한 팀이 득점하거나, 주어진 시간이 다 소비되면 수행을 끝낸다. 경기 중에 같은 방안에 n대의 로봇과 공이 존재하는 경우 처리 방법은 보통 두 가지이다. 무작위로 아무나 차지하게 하거나 둘 다 차지하지 못하게 하는 방법이그것이다. 본 연구에서는 각각의 로봇이 공을 차지할 확률은 1/n로 같다고 가정하고 모의실험 하였다. 실제 로봇 축구에서 드리블하는 로봇이 다른 로봇과의 접촉시 공의 소유에 있어서 우위를 점하기 힘든 것이 사실이기 때문이다. 따라서, 같은 규칙표와 초기 조건을 가지고도 다른 결과가 나올 수 있으며 이는 뒤에 설명하게 될 적합도 함수가 수 차례의 모의실험을 포함하도록 한 이유이다.

3.3 문제의 정의 및 유전 프로그래밍의 적용

모의 실험에 사용한 SS 모델은 그림 4에 보인 것과 같다. 즉, 5(가로) × 6(세로)개의 방으로 이루어져 있다. 각 로봇의 초기 위치도 그림 4에 보인바와 같다. 팀 B의 행동 양식은 미리 주어져 있는 것으로 생각했다. 양팀의 행동 전략을 동시에 찾아나가도록 하는 방법을 찾는 것이 쉽지 않기 때문이다. 팀 A가 득점할 수 있는 규칙표를 유전 프로그래밍을 사용하여 찾아내는 것이 목표이다. 다음의 두 가지 경우를 모의 실험 하였다.

문제 1: 로봇 B1과 B2는 그림 4에서의 위치로 게임 내내 고정되어 있어 단순히 장애물로 작용한다.

문제 2: 공의 초기 가로 위치는 왼쪽 끝이며 B1은 골포스트 앞에 고정되어 있고, B2는 공을 따라다니도록 하여 상대방의 드리블과 킥을 방해하도록 하였다.

유전 프로그래밍을 적용하기 위해서는 적합도 함수를 정의해야 한다. 공을 오래 가지고 있을수록, 또 득점할수록 커지도록 정의된 적합도는 다음과 같다.

$$F=\sum_{i=1}^{n_{iter}} F'(i) \quad (2)$$

$$F'(i)=F_0+\sum_{t=0}^{t_{final}}(f_{goal-A}(t)+f_{possess-A}(t)+f_{goal-B}(t)+f_{possess-B}(t)) \quad (3)$$

여기서, i는 모의 실험 번호를 나타낸다. n_{iter}는 5를 t_{final}은 30을 사용했다. 여기서, n_{iter}는 해 하나를 평가할 때 수행되는 회수를 의미함을 주목하라. F_0는 적합도가 양이 되도록 하는 상수로서 400을 사용했다. 다른 함수들은 다음과 같다.

$$f_{goal-A}(t)=\begin{cases}1000, & \text{팀 } A\text{가 시간 } t\text{에서 득점할 때}\\ 0, & \text{그렇지 않은 경우}\end{cases} \quad (4)$$

$$f_{possess-A}(t)=\begin{cases}10, & \text{팀 } A\text{가 시간 } t\text{에서 공을 가지고있을때}\\ 0, & \text{그렇지 않은 경우}\end{cases} \quad (5)$$

$$f_{goal-B}(t)=\begin{cases}-90, & \text{팀 } B\text{ 시간 } t\text{에서 득점할 때}\\ 0, & \text{그렇지 않은 경우}\end{cases} \quad (6)$$

$$f_{possess-B}(t)=\begin{cases}-10, & \text{팀 } B\text{가 시간 } t\text{에서 공을 가지고 있을때}\\ 0, & \text{그렇지 않은 경우}\end{cases} \quad (7)$$

교차변이 연산자 적용 확률은 0.8, 돌연변이 연산자 적용 확률은 0.1 그리고, 집단의 개체 수는 50으로하여 300세대까지 모의 실험 하였다.

3.4 모의 실험 결과 및 분석

그림 6, 7에 문제 1에 대한 결과를, 그림 8, 9에 문제 2에 대한 결과를 각각 나타내었다. 각각의 그림은 세대수에 따른 가장 높은 적합도를 나타내고 있다. 유전 프로그래밍 자체가 확률적 기법이므로, 각 문제에 대하여 다섯 번씩 모의 실험 하였으며 그림 6, 8에는 그 중 가장 좋은 결과를, 그림 7, 9에는 평균값을 나타내었다. 앞에서 정의한 적합도 함수를 고려할 때, 적합도가 대략 1000 이상이 되는 시점은 팀 A가 득점할 수 있는 규칙을 얻은 것을 나타낸다. 문제 1은 약 10세대만에, 좀 더 어려운 문제인 문제 2의 경우는 가장 좋은 결과에선 약 100세대 후에 그리고, 평균적으로는 약 190세대 후에야 처음으로 팀 A가 득점하는 것을 볼 수 있다. 가능한 규칙은 총 3,300개이지만 각 모의실험의 iteration이 30이고 A 팀의 선수가 2명이므로 평균적으로 60개 정도의 조건-출력 규칙이 사용될 수 있으며 결과적으로 얻어진 규칙 표들을 분석한 결과, 평균적으로 문제 1의 경우는 15개의 조건-출력 규칙이, 문제 2는 20개 정도의 규칙만이 해를 구성하였다.

유전 프로그래밍을 적용할 때 고려되지 않았음에도 결과

에서는 드리블, 패스 그리고 킥 등의 상호 협력 양태를 보였다. 같은 방안에 여러 대의 로봇이 있을 경우 확률적으로 공을 차지하는 규칙으로 인하여, 득점을 하기까지 좀 더 많은 드리블 및 패스가 필요한 문제 2의 경우가 세대수에 따른 적합도 변화가 심함을 볼 수 있다.

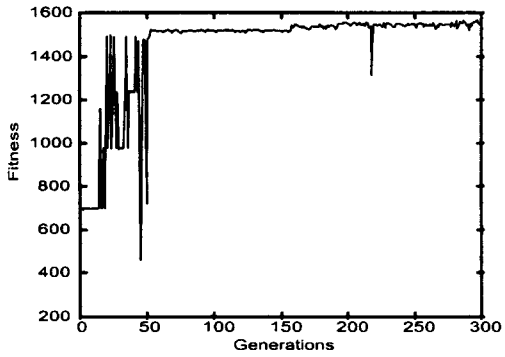


그림 6 문제 1 의 결과 (best)
Fig. 6 The result from the problem 1 (best)

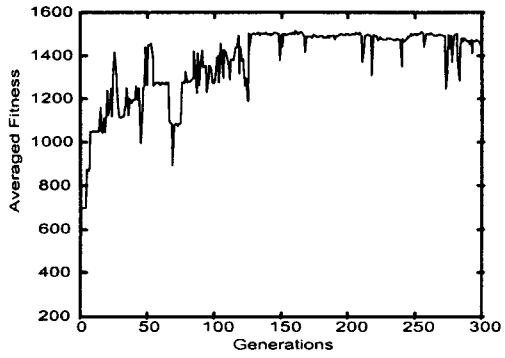


그림 7 문제 1 의 결과 (평균값)
Fig. 7 The result from the problem 1 (average)

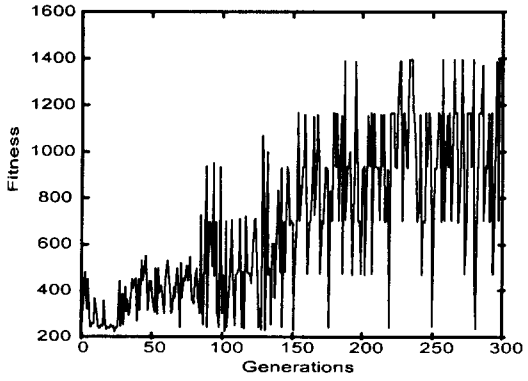


그림 8 문제 2 의 결과 (best)
Fig. 8 The result from the problem 2 (best)

그림 6~9에 보여진 그래프들의 들쭉날쭉함 자체에는 큰 의미가 없다. 하지만, 세대가 거듭됨에 따라 전체적으로 증가하는 양상이 중요하다. 유전 프로그래밍에 의하여 로봇들의 바람직한 행동을 유도하는 규칙들이 학습되고 있다는 증거이기 때문이다. 여기서, 본 응용 문제에 유전 프로그래밍을 이용함에 있어서 어떠한 사전 정보도 이용하지 않았다는 사실을 주목할 필요가 있다. 진화 연산 기법의 일반적인 장점이기도 하지만, 미지의 환경에서 원하는 작업에 대한 평가 함수만 존재할 때에도 구체적인 작업 방법을 알아내는 일은 기존의 해석적 접근 방법을 쓸 수 없기 때문에 매우 유용하다.

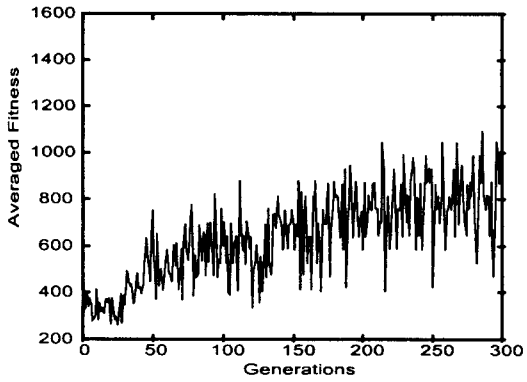


그림 9 문제 2 의 결과 (평균값)
Fig. 9 The result from the problem 2 (average)

3.5 신경회로망 제어기의 설계

앞에서 설명했듯이 규칙 표를 신경회로망에 학습시키면 신경회로망 제어기의 구성이 가능하게 된다. 본 연구에서는 그림 10과 같은 feed-forward 형태의 신경회로망을 이용하였다.

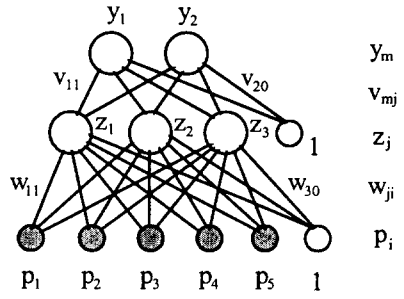


그림 10 사용된 신경회로망
Fig. 10 Neural network used in the paper

여기에서, 5개의 입력노드는 각각 C, L, R, T, B 를 나타낸다. 은닉층의 노드는 3개로 정했으며 출력층의 첫 번째 노드는 'move' 또는 'dribble' 인지 아니면 'kick' 인지를 출력하고, 두 번째 노드는 방향을 출력하게 된다.

성공적인 규칙표 하나를 택해 해의 궤적에 영향을 미친 규칙만을 추려내어 학습 데이터로 사용하였다. 비교적 구조가 간단한 feed-forward 신경망 학습에 매우 빠르게 동작하는 Levenberg-Marquardt 학습 방법을 사용하였다.

연속적인 입력과 출력을 가지며 다양한 입력·출력 매핑을 장점으로 하는 신경회로망을 규칙 기반 제어기에서와 같은 환경에서 적용하는 것은 별 의미가 없을 것이다. 따라서, 신경회로망의 장점을 살릴 수 있도록 cellular 환경이 아닌 연속적인 환경에 가까운 환경에 적용해보고자 한다. 방의 크기를 줄이는 것(즉, 방의 개수를 많아지게)이 한 방법이 될 것이다. 신경회로망의 내삽(interpolation) 능력에 의해 학습 데이터에 속해 있지 않았던 상황에서도 적절한 행동을 명령할 수 있으리란 기대를 할 수 있다.

모의실험 결과 연속적인 공간에서 이상적으로 동작하는 신경회로망 제어기를 구성할 수는 없었지만, 원래 환경보다는 복잡한 환경에서 동작함으로써 그 가능성은 엿볼 수 있었다. 그림 11은 방의 크기가 반으로 줄어든, 즉, 방의 개수는 4배가 된 환경 하에서의 문제 1에 대한 각 로봇의 궤적을 나타낸다. 변화된 방의 개수로 인하여 원래의 규칙표는 적용될 수 없다. 그림 11에서 각 기호의 의미는 다음과 같다. '◇'는 팀 B, 'X'는 팀 A 그리고, 'O'는 공을 나타낸다. 원래의 궤적과 일치하진 않지만 유사한 궤적을 보였으며, 더 복잡해진 환경 하에서도 신경회로망 제어기에 의해 팀 A의 로봇이 성공적으로 제어될 수 있음을 확인할 수 있다.

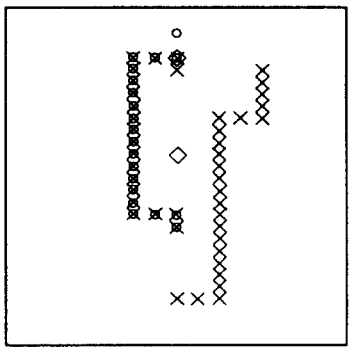


그림 11 신경회로망 제어기를 이용해서 얻은 로봇의 궤적
Fig. 11 Trajectories of the robot controlled by neural network

4. 결론

인공지능 언어인 LISP을 이용하여 유전 프로그래밍 방법을 프로그래밍 하였으며, 이동 로봇이 간단한 축구 경기를 하는 문제에 적용하여 규칙 기반 제어기를 설계함으로써 미지의 환경에서 상호 협력하는 로봇 제어기를 사전 정보 없이 유전 프로그래밍을 통해서 얻을 수 있음을 확인하였다.

유전 프로그래밍을 통해 얻어진 규칙 기반 제어기에서 규칙을 추출하여 학습 데이터로 이용함으로써 신경망 제어기를 구성할 수 있음을 확인하였다. 신경망 제어기는 규칙 기반 제어기보다 더 복잡한 환경 하에서의 제어를 가능하게 해준다.

보다 많은 개체를 포함하는 경우에 대한 실험 및 연속적인 환경에서 성공적으로 동작하는 신경망 제어기의 구성 방법은 추후 연구되어야 할 것이다.

감사의 글

본 연구는 1996년도 산학협동재단의 학술연구비 지원에 의하여 연구되었습니다.

참 고 문 헌

[1] D. B. Fogel, Tutorial S-5: Evolutionary Computation, *IEEE Int. Conf. on Robotics and Automation*, 1994

[2] K. Kristinsson and G. A. Dumont, "System Identification and Control Using Genetic Algorithms," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 5, pp. 1033-1046, Sep., 1992

[3] Y. Ichikawa and T. Sawa, "Neural Network Application for Direct Feedback Controllers," *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 224-231, Mar., 1992

[4] I. K. Jeong and J. J. Lee, "A Modified Genetic Algorithm for Neurocontrollers," *IEEE Int. Conf. on Evolutionary Computation*, pp. 306-311, Dec., 1995

[5] C. L. Karr and E. J. Gentry, "Fuzzy Control of pH Using Genetic Algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 1, no. 1, pp. 46-53, Feb., 1993

[6] W. R. Hwang and W. E. Thompson, "Design of Intelligent Fuzzy Logic Controllers Using Genetic Algorithms," Proceedings of the Third *IEEE Int. Conf. on Fuzzy Systems*, pp. 1383-1388, 1994

[7] J. R. Koza, Genetic programming, Cambridge, MA, MIT Press, 1993

[8] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA, Addison-Wesley, 1989

[9] L. Davis, Handbook of Genetic Algorithms, NewYork, Van Nostrand Reinhold, 1991

[10] WWW site, http://www.genetic-programming.org/

[11] M. J. Patel and V. Maniezzo, "NN's and GA's: Evolving co-operative behavior in adaptive learning agents," *IEEE Int. Conf. on Evolutionary Computation*, pp. 290-295, 1994

[12] J. D. Lohn and J. A. Regiia, "Discovery of Self-Replicating Structures Using a Genetic Algorithm," *IEEE Int. Conf. on Evolutionary Computation*, pp. 678-683, Dec., 1995

[13] W. Pedrycz, Fuzzy control and fuzzy systems, New York, John Wiley & Sons, 1993

[14] H. Nagahashi et al., "Competetion and Mutualism in a Simulation of Adaptive Artificial Organisms," *IEEE Int. Conf. on Evolutionary Computation*, pp. 695-700, Dec., 1995

[15] M. L. Wong and K. S. Leung, "Combining Genetic Programming and Inductive Logic Programming using Logic Grammars," *IEEE Int. Conf. on Evolutionary Computation*, pp. 733-736, Dec., 1995

[16] J. D. Schaffer et al., "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art," *Int. Workshop on Combinations of Genetic Algorithms and Neural Networks*, pp. 1-37, June, 1992

[17] K. A. De Jong, "Genetic Algorithms are NOT Function Optimizers", *Foundations of Genetic Algorithms 2*, pp. 5-17, 1992.

[18] I. K. Jeong and J. J. Lee, "Evolving Cooperative Mobile Robots Using a Modified Genetic Algorithm," *Robotics and Autonomous Systems*, 21, pp. 197-205, 1997

저 자 소 개

정 일 권 (丁 一 權)

1970년 8월 14일 생. 1992년 한국과학기술원 전기및전자공학과 졸업. 1994년 동 대학원 전기및전자공학과 졸업(공학석사). 1999년 동 대학원 전기및전자공학과 졸업(공박). 현재 한국전자통신연구원 가상현실연구센터 선임연구원.

Tel : (042) 860-1230, Fax : (042) 860-1051,

E-mail : jik@etri.re.kr

이 주 장 (李 柱 張)

1948년 11월 14일 생. 1973년 서울대학교 전기공학과 졸업. 1977년 동 대학원 전기공학과 졸업(공학석사). 1984년 Univ. of Wisconsin 졸업(공박). 1978~ 1980 미국 G.T.E. Automatic Electric Co. Project Engineer. 현재 한국과학기술원 전기 및 전자공학과 교수. 제어·자동화·시스템공학회 로보틱스 및 응용연구회 위원장 및 대전 지부장

Tel : (042) 869-3432, Fax : (042) 869-3410,

E-mail : jjlee@ee.kaist.ac.kr