DOI: 10.3745/KIPSTB.2008.15-B.3.253

유전 프로그래밍을 이용한 추격-회피 문제에서의 게임 에이전트 학습

권 오 광[†] · 박 종 구[††]

요 약

최근의 게임 플레이어들은 단순한 반복적인 조작을 벗어나 복잡한 환경 하에서 다양한 전략과 전술을 구사하여야 하는 게임을 요구하고 있다. 이러한 환경에서 게임 캐릭터를 학습시키기 위해 다양한 인공지능 기법들이 제안되었으며, 최근에는 신경망과 유전 알고리즘을 이용한 학습 방법이 연구되고 있다. 본 논문에서는 게임이론에서 널리 사용되는 추격-회피 전략의 학습을 위해 유전 프로그래밍(GP)을 사용하였다. 제안된 유전 프로그래밍은 신경망과 같은 기존의 방법에 비해 수행 속도가 빠르고, 학습의 결과를 직관적으로 이해할 수 있으며, 진화된 염색체를 추론 규칙으로 변환 가능하므로 호환성이 높다는 장점을 가지고 있다.

키워드 : 인공지능, 결정 트리, 유전 프로그래밍, 추격-회피 문제

Game Agent Learning with Genetic Programming in Pursuit-Evasion Problem

O-Kyang Kwon[†] · Jong-Koo Park[††]

ABSTRACT

Recently, game players want new game requiring more various tactics and strategies in the complex environment beyond simple and repetitive play. Various artificial intelligence techniques have been suggested to make the game characters learn within this environment, and the recent researches include the neural network and the genetic algorithm. The Genetic programming(GP) has been used in this study for learning strategy of the agent in the pursuit-evasion problem which is used widely in the game theories. The suggested GP algorithm is faster than the existing algorithm such as neural network, it can be understood instinctively, and it has high adaptability since the evolving chromosomes can be transformed to the reasoning rules

Key Words : Artificial Intelligence, Decision Tree, Genetic Programming, Pursuit-Evasion Problem

1. 서 론

최근의 게임들은 단순한 순발력에 의존한 조작이나 반복적인 플레이보다는 복잡한 환경에서 다양한 전략과 전술을 구사하도록 설계되는 경향이 강하다. 이에 따라 게임 캐릭터들에게 지능을 부여할 필요성은 점점 증가하고 있다.[1] 초기에는 단순한 규칙 기반 추론이 사용되었지만, 최근 들어서는 신경망과 같은 인공지능 기법이 사용되는 경우도 늘어나고 있다. [2-7]

본 논문에서는 유전 프로그래밍을 이용해 캐릭터를 학습시키는 방법을 제안한다. 제안된 방법은 게임 캐릭터에 적용될 때 두 가지 이점을 가진다. 첫째로, 유전 프로그래밍으로 생성된 트리는 신경망보다 수행 속도가 빠르다. 신경망의 처리에는 여러 번의 곱셈 연산이 들어가는 데 반해, 제안된 방법은 if-then 구문만으로 구성할 수 있기 때문이다. 둘째, 아직까지 실제 게임에서는 규칙기반 추론을 사용하는 곳이 대부분이다. 제안된 방법으로 생성된 유전자는 추론 규칙으로 변환이 가능하므로, 규칙기반 추론을 사용하는 시스템에 그대로 적용할 수 있다.

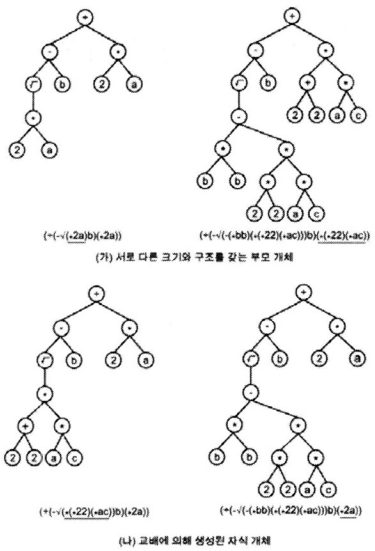
제안된 방법이 유용함을 입증하기 위해, 게임 이론에서 널리 사용되는 추격-회피 문제를 유전 프로그래밍으로 학습하는 실험을 진행하였다.

2. 관련 연구

2.1 유전 프로그래밍

존 코자(John Koza)에 의해 제안된 유전 프로그래밍은 컴퓨터가 인간에 의해 프로그램 되지 않은 상태에서 스스로

† 정 회 원 : 성균관대학교 컴퓨터공학부 석사과정 졸업
†† 정 회 원 : 성균관대학교 정보통신공학부 교수
논문접수 : 2007년 11월 12일
수 정 일 : 1차 2007년 12월 27일, 2차 2008년 1월 23일
심사완료 : 2008년 1월 27일



(그림 1) 서로 다른 크기와 구조를 가지는 유전 프로그래밍의 염색체들

문제 해결을 위한 프로그램을 생성하는 기법이다. 유전 프로그래밍은 유전 알고리즘에 기반을 두고 있으며, 염색체로서 이진수나 실수의 배열 대신 트리 구조를 사용한다.

트리의 내부 노드는 자식 노드의 값을 인수로 가지는 연산자 혹은 함수로 사용되며, 단말 노드는 변수나 상수로 사용된다. 이는 프로그래밍 언어인 리습(LISP)에서 사용되는 형태이다.

유전 프로그래밍에서의 교배는 트리의 한 점에서 하위 트리를 서로 교환하는 형태로 이루어진다. 그러므로 교배를 통해 발생된 자식 개체는 (그림 1)과 같이 부모와 다른 크기와 구조를 가지게 된다.[8]

유전 프로그래밍에서는 각 개체가 시간이 지남에 따라 커지는 경향이 있다[9]. 이러한 현상을 부풀림(Bloat)라고 하며, 실행 시간을 늘리고 분석을 어렵게 하는 요인이 된다.

부풀림을 줄이기 위한 방법으로, 트리의 노드 수를 적합도 함수의 인자로 넣어 평가하는 방법이 있다. 이 경우 적합도 함수는 다음과 같이 수정된다.

$$fitness = f(Error + Size(t_i) * a)$$

f는 적합도 함수이며 t_i는 i번째의 트리, a는 트리의 크기가 적합도에 영향을 주는 비중을 결정하는 가중치 값이다.

2.2 추격-회피 문제

추격-회피 문제는 수많은 다양한 변형이 존재하지만, 기본적인 형태는 그래프의 노드 상에 추격자와 도망자가 위치하는 구조이다. 추격자와 도망자는 각각 자신의 차례에 그래프의 인접 노드로 이동할 수 있으며, 추격자가 도망자와 같은 노드에 위치하게 되면 도망자는 붙잡혀 그래프에서 제거된다.

추격-회피는 게임 이론에서 다루는 주요 연구 테마 중 하나이며, 여기에는 몇 가지 이유가 있다. 첫째, 추격과 회피는 서로간의 이익의 충돌에 의해 생기는 일반적인 문제이다. 둘째, 동적이고 확률적이며 연속된 공간과 연속된 시간의 게임이기 때문에 해결하기 어렵다. 셋째, 하나의 공간을 두고 상호 진화가 일어나므로 과학적으로 흥미롭다. 넷째, 행동생물학과 신경 동물 행동학, 게임 이론의 세 가지 과학적 연구에서 주목받아 왔다. 마지막으로, 많은 과학적 의미를 지니고 있으며 실제적 적용이 가능하다.[10]

추적-회피 문제는 강화 학습이나 유전 알고리즘을 통한 다양한 연구가 이루어져 왔다.[11-13]

2.3 결정 트리(Decision tree)

결정 트리는 결정 규칙(decision rule)을 트리 구조로 도표화하여 분류 및 예측을 수행하는 분석 방법이다. 이 방법은 추론 규칙이 트리 형태에 의해 표현되기 때문에, 다른 방법에 비해 연구자가 분석 결과를 이해하기 편리하다는 장점을 가지고 있다.

결정 트리 알고리즘은 하향(top-down)방식으로, 루트 노드에서 사례들을 가장 잘 분리하는 특성을 선택하고 그 특성의 값(또는 범주)에 따라 사례들을 분류한다. 단말 노드에 속하는 모든 사례들이 같은 클래스 값을 가질 때까지 이 과정을 반복함으로서 결정 트리를 생성하게 된다. 이러한 분류 알고리즘으로는 CART(Classification And Regression Trees)와 ID3 등이 있다.

하향 방식의 결정 나무 생성 알고리즘에서는 각 분류마디에서 최적인 특성을 선택하여 사례들을 분류하기 때문에 국소 최적(local optima) 문제가 발생할 수 있다.[14] 또한 학습 과정에서 과분류가 이루어질 경우 잉여 특성(redundancy feature)이 생겨 생성된 결정 트리를 이해하기 어렵게 만들고, 결정 트리의 전체 정확성을 손상시키게 된다.[15] 이러한 문제를 해결하기 위해 결정 트리를 유전 프로그래밍으로 학습시키는 방법이 연구되었다.[16-17]

3. GP를 이용한 추론 규칙 생성

본 논문에서는 if-then문으로 변환 가능한 추론 규칙을 생성하기 위해 유전 프로그래밍을 사용한다. 결정 트리를 학습하는 기존의 알고리즘은 주로 자료 분류에 사용되는 방법으로, 환경이 실시간으로 변화하는 상황에는 적합하지 않다.

제안하는 방법은 유전 프로그래밍의 연산자로서 if-then문만을 사용한다. 입력값과 출력값으로 각각 1차원의 2진수 배열이 주어지며, 각 비단말 노드는 입력 배열상의 특정 위치를 지정한다. 단말 노드의 값은 출력 배열상의 특정 위치이다. 결과적으로, 생성된 트리는 결정 트리의 형태를 취하게 된다.

3.1. 기본 구조

제안된 구조는 추론 규칙의 역할을 하는 프로그램, 즉 염색체와 입력 배열 및 출력 배열로 이루어진다.

3.1.1 입력/출력 배열

입력 및 출력 배열은 프로그램을 실행하기 위한 입출력 값을 저장하는 공간이다. 입력/출력 배열은 2진수의 1차원 배열로 구성되며, 프로그램(염색체)은 입력 배열의 값을 읽어들여 추론을 수행한 후, 그 결과를 출력 배열에 저장한다.

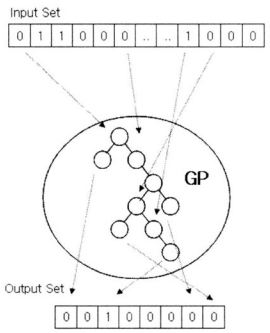
3.1.2 염색체 구조

각 염색체는 결정 트리 형태의 구조를 지닌다. 각각의 비단말 노드는 입력 배열의 특정 위치의 값이 참인지 거짓인지를 판단하고, 그 결과가 참일 경우 오른쪽 자식 노드로, 거짓일 경우 왼쪽 자식 노드로 분기한다. 이는 입력 배열의 특정값을 인자로 받아 참인지의 여부를 판단하는 함수로 볼 수 있다.

트리의 단말 노드는 추론의 결과값을 저장하고 있다. 여기서 결과값은, 출력 배열의 위치이다. 추론이 반복되어 단말 노드까지 도착했을 때, 단말 노드는 자신이 가지고 있는 결과값에 해당하는 출력 배열상의 비트를 반전시킨다.

결과적으로, 제안된 유전 프로그램은 (그림 2)와 같이 2진수 입력 배열을 입력받아 출력 배열의 특정 비트를 1로 설정하는 기능을 하게 된다. 프로그램의 수행 과정은 <표 1>과 같다.

사용하는 연산자가 한 종류뿐이므로, 비단말 노드는 함수의 타입을 저장할 필요가 없다. 각 비단말 노드에는 인자, 즉 입력 배열의 위치를 가리키는 정수값만이 저장된다. 마찬가지로, 각 단말 노드에는 출력 배열의 위치를 가리키는 정수값이 저장된다.



(그림 2) 제안된 유전 프로그래밍의 기본 구조

〈표 1〉 프로그램의 수행 과정

1. 루트 노드를 현재 노드로 선택한다.
2. 현재 노드가 단말 노드인지를 검사한다.
3. 단말 노드가 아닐 경우, 노드가 가리키는 Input set상의 위치의 값을 판별해, 1(true)일 경우 오른쪽 자식 노드, 0(false)일 경우 왼쪽 자식 노드로 이동한다. 해당하는 자식 노드가 없을 경우 종료한다.
4. 단말 노드일 경우, 노드가 가리키는 output set상의 위치의 값을 1로 바꾸고 종료한다.
5. 2로 이동한다.

3.2 추론 규칙의 분석

제안된 구조는 두 개의 자식 노드를 가지는 결정 트리로 구성되며, 이는 이진 트리의 형태를 가진다.

이 구조를 이용해 AND, OR, NOT 등의 논리 회로를 구성할 수 있으며, 이는 제안된 구조가 컴퓨터로 처리 가능한 범용적인 문제에 사용될 수 있음을 의미한다.

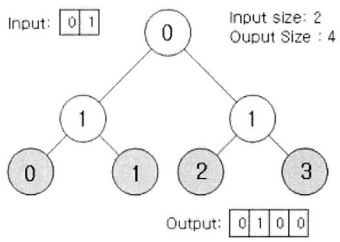
(그림 3)은 1×4 디멀티플렉서를 염색체 형태로 나타낸 것이다.

이 때 수행 과정은 다음과 같다.

① 루트 노드에서 입력 배열의 0번째 위치가 참인지를 확인한다.
② 입력 배열의 0번째 값은 0이므로, 왼쪽 자식 노드로 분기한다.
③ 입력 배열의 1번째 위치가 참인지를 확인한다.
④ 해당 값이 1이므로, 오른쪽 노드로 분기한다.
⑤ 분기한 노드가 단말 노드이며, 값이 1이므로 출력 배열에서 해당하는 위치의 값을 1로 설정한다.

제안된 염색체 구조는 if-then 형태와 1:1로 대응시킬 수 있다. (그림 3)의 염색체를 if-then 형태로 바꾼 결과는 <표 2>와 같다.

이렇게 만들어진 규칙은 FSM(Finite state Machine)이나 간단한 if-then문의 집합으로 캐릭터의 인공지능을 구현하는 시스템에서도 별도의 작업 없이 곧바로 사용할 수 있다. 기존의 게임 중 상당수가 이러한 구조임을 생각해볼 때, 이는 강력한 장점이 될 수 있다.



(그림 3) 1x4 디멀티플렉서의 구조

〈표 2〉 if-then 구조

```
If(input_set[0] == 1) then
    If(input_set[1] == 1) then
        output_set[3] = 1
    else
        output_set[2] = 1
else
    If(input_set[1] == 1) then
        output_set[1] = 1
    else
        output_set[0] = 1
```

3.3 진화 과정

생성된 추론 규칙은 일반적인 GP의 진화 방법을 사용하며, 진화 연산자로서 교배(crossover)와 돌연변이(mutation)를 사용한다.

염색체의 교배는 두 부모 사이에서 랜덤하게 선택된 노드를 기준으로 노드의 하위 트리를 교환함으로서 이루어진다. 이러한 교배의 결과로 염색체의 크기와 구조가 변화할 수 있다.

돌연변이는 일정 확률로 노드에 저장된 값을 임의의 다른 값으로 바꾸는 형태로 이루어진다. 이 때, 입력 배열과 출력 배열의 크기가 다를 수 있기 때문에 단말 노드와 비단말 노드의 돌연변이는 다르게 적용되어야 한다.

진화에 필요한 적합도는 추격-회피 문제에서 얻은 점수로 결정되며, 부풀림 문제를 해결하기 위해 트리의 크기가 적합도에 추가된다.

3.4 수행 속도 비교

뉴럴 네트워크의 역전파 학습과 비교했을 때, 제안된 GP 구조는 직관적이며 높은 호환성을 지닌다는 것 외에도 수행 속도면에서 이점을 가질 수 있다.

NN과 GP의 수행 속도는 다음과 같다.

역전파: $n_I * n_{H_1} + \sum_{i=1}^{k-1}(n_{H_i} * n_{H_{(i+1)}}) + n_{H_k} * n_O$회의 곱셈이 필요하므로, 은닉층이 1개일 경우 $n_I + n_O > n_H$일 때 수행시간은 $O(n^2)$이라고 볼 수 있다.

GP: 트리의 루트에서 단말 노드까지의 이동이므로 worst case일 때 $O(n)$, average case는 $O(\log n)$이다.

역전파 알고리즘에서 n은 은닉 노드의 개수이고 제안된 GP 알고리즘의 경우는 전체 노드 숫자이지만, 은닉 노드가 10개 이상일 경우 트리의 노드 수가 상당히 크다고 해도 수행 속도면에서 제안된 방법이 유리함을 알 수 있다.

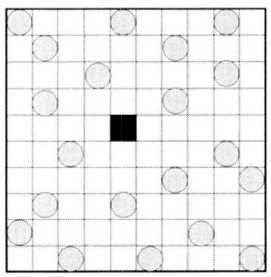
4. 실험 설계

4.1 실험 환경

실험은 (그림 4)와 같은 형태의 10×10 크기의 격자 공간에서 진행된다. 공간은 상하와 좌우가 연결되어 있으므로, 한 방향으로 계속 이동하다 보면 원래의 위치로 돌아오게 된다.

격자 공간 내의 특정 위치에는 하나의 추격자 또는 도망자만이 존재할 수 있다. 도망자는 자신의 주위 4방향을 검색해서 추격자가 없는 방향으로 이동하고, 이동할 곳이 없게 되면 그 자리에 멈추어 있게 된다. 추격자는 유전 프로그래밍을 이용해 학습된 행동을 선택한다.

추격자 주위의 4방향 중 한 곳에 도망자가 있을 때 추격자가 공격 행동을 선택하면 1점의 점수를 얻는다.



(그림 4) 격자 공간

4.2 유전 프로그래밍 설계

유전자는 3절에서 제안한 추론 규칙 생성을 위한 유전 프로그래밍 구조를 사용한다. 실험 환경에서 유전 프로그래밍의 입력값은 캐릭터 주위의 5×5 공간에 추격자 혹은 도망자가 있는지에 대한 정보이다. 멀티 에이전트 환경에서 추적자는 근처에 있는 동료 추적자의 위치를 입력받으며, 이를 통해 상호 협력을 학습할 수 있다.

입력값으로 자신이 위치한 공간의 정보는 필요하지 않으므로, 입력값의 개수는 24×2=48개가 된다. 출력값은 4방향으로의 이동과 공격 명령을 위한 1개를 포함해 총 5개가 사용된다.

학습에 사용된 파라미터는 아래와 같다.

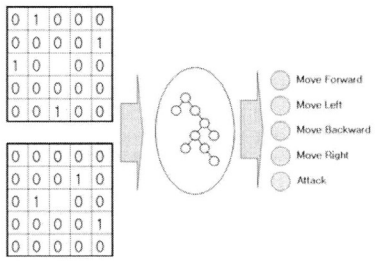
개체의 적합도를 평가하기 위한 공식은 다음과 같다.

$$Fit^i = \sum_{t=0}^{T}(f_1^i(t) + f_2^i(t)) - a \cdot size$$

이 때 f_1^i은 i번째 추적자가 도망자와 인접했을 때(상하좌우 4방향 중 한 곳에 위치할 때) 1, 아닐 경우 0이며, f_2^i는 I번째 추적자가 도망자와 인접한 상태에서 공격 행동을 취했을 때 2, 아닐 경우 0의 값을 가진다. T는 실험 시간인 20이다. size는 트리의 크기이며, a는 가중치 값이다. a값을 0으로 두는 경우와 0.001로 두는 두 가지 경우에 대해 실험이 진행되었다.

〈표 3〉 진화 파라미터

진행 시간	20턴
추적자의 수	20
개체수	100개
돌연변이	0.5%
선택압(Selection Pressure)	4
시작시 트리 크기(노드수)	300
학습 세대	최대 1000



(그림 5) 프로그램 구조

5. 결과 분석 및 비교

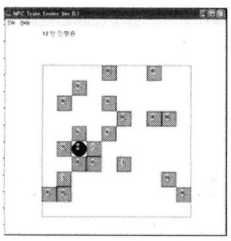
5.1 실험 결과

제안된 유전 프로그래밍과 실험 환경을 시뮬레이션 하는 프로그램을 제작하여 실험을 진행하였다.

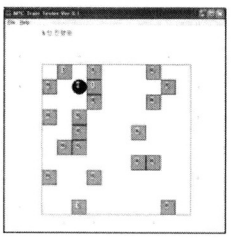
(그림 6)과 같이 추적자(사각형)가 도망자(원)를 네 방향에서 완전히 둘러싸면 도망자는 더 이상 도망칠 수 없게 되며, 추적자는 이 상태에서 가장 높은 점수를 얻을 수 있다.

도망자는 추적자가 없는 방향으로 계속해서 이동하므로, 네 방향에서 도망자를 둘러싸기 위해서는 추적자들 간의 협력과 통신이 필요하다. 시뮬레이션에서 추적자는 도망자가 일정 범위 안에 들어올 경우 이동 경로를 막으면서 포위하는 형태의 전략을 학습했으며, 이는 추적자들 간에 협동이 이루어지고 있음을 뜻한다.

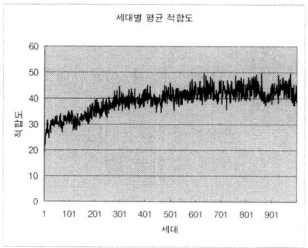
1천 세대 동안 학습한 결과, 세대별 평균 적합도는 (그림 8)과 같다.



(그림 6) 시뮬레이터 실행 화면



(그림 7) 도망자를 몰아넣고 있는 추적자



(그림 8) 세대별 평균 적합도

5.2. 기존 알고리즘과의 비교

제안된 방법을 동일한 환경에서 신경망을 유전 알고리즘으로 학습한 결과와 비교하였다. 역전파(backpropagation) 알고리즘은 본 실험 환경과 같은 복잡한 비선형 문제에는 적합하지 않으며, 유전 알고리즘과 역전파의 비교는 이미 다루어진 문제이기 때문에 제외하였다.[18]

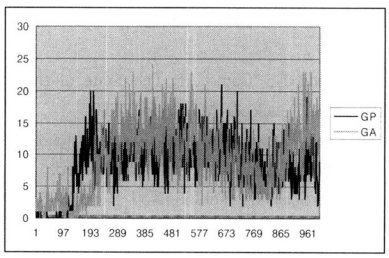
실험에서 적합도는 반드시 포획 횟수와 비례하는 것은 아니며, 적합도보다는 실제로 포획에 성공한 횟수가 학습의 성과를 정확히 반영한다.

(그림 9)는 각 세대에서 가장 우수한 개체의 포획 성공 횟수 그래프이다. 결과의 질에서 두 알고리즘은 큰 차이가 없음을 알 수 있다.

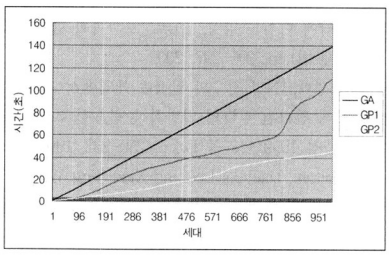
두 알고리즘이 1천 세대에 도달하는 데 걸린 시간은 (그림 10)과 같다.

학습 과정에서 적합도를 구하기 위해 실험을 반복적으로 진행하기 때문에, 알고리즘의 수행 속도는 결과적으로 학습 속도와 직결된다. GP1은 적합도 함수의 인자로 트리의 크기를 포함시키지 않았을 경우이며, GP2는 크기가 포함된 경우이다. 신경망을 유전 알고리즘으로 학습한 경우에 비해 제안된 방법이 수행 시간이 짧음을 알 수 있다. 특히, 트리의 크기가 적절하게 제어될 때 수행 속도의 격차는 더욱 커진다.

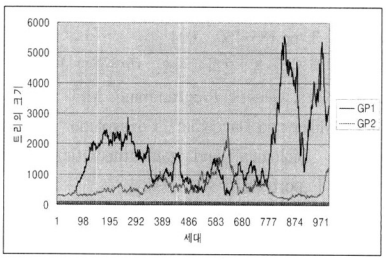
유전 프로그래밍에서는 진화 과정에서 염색체의 크기가 계속 변화하기 때문에 수행 시간 역시 동적으로 바뀐다. (그림 11)은 트리의 크기를 적합도 함수의 인자로 포함시켰을 경우와 포함시키지 않았을 경우의 세대별 트리의 평균 크기이다. (그림 10)과 비교해 보면, 트리의 크기가 학습 시간에 영향을 주고 있음을 알 수 있다.



(그림 9) 세대별 포획 성공 횟수



(그림 10) 학습 시간



(그림 11) 트리의 세대별 평균 크기

6. 결론 및 향후 연구

실험을 통해 유전 프로그래밍으로 추론 규칙을 생성하고 학습시키는 것이 가능하며, 속도면에서 기존의 방법에 비해 우수함을 알 수 있었다.

제안된 방법은 기존의 신경망에 비해 결과를 분석하고 기존의 응용 프로그램에 적용시키기에 유리하며, 수행 속도면에서 장점을 가진다.

본 논문에서는 추격-회피 문제를 다루었지만, 제안된 방법은 조건식으로 구성되는 추론 규칙을 생성하는 데 폭넓게 사용될 가능성을 가지고 있다.

향후 과제는 실제 게임과 유사한 모델을 이용해 실험을 진행하는 것이다. 게임 개발에서 이와 같은 인공지능 기법의 사용은 게임 개발 시간의 상당 부분을 차지하는 NPC들의 행동 패턴 설계나 밸런스 작업에 도움을 주고, 게임 환경이 동적으로 변화하는 경우에도 캐릭터가 그에 맞춰서 자동적으로 학습을 진행할 수 있는 환경을 제공할 수 있을 것이다.

참 고 문 헌

[1] Steve Woodcock, "Game AI: The State of the Industry 2002," Game Developer Magazine, 2002.

[2] Malrey Lee, "Evolution of behaviors in autonomous robot using artificial neural network and genetic algorithm," Information Sciences, Vol.155, No.1-2, pp.43-60, 2003.

[3] 조병헌, 정성훈, 성영락, 오하령, "신경망 지능 캐릭터의 게임 환경 변화에 대한 적응 방법," 전자공학회논문지-CI, Vol.42, No.3, pp.17-28, 2005.

[4] 박사준, 김성환, 김기태, "인공 유기체의 학습 행동이 게임 캐릭터의 전략에 미치는 영향," 한국정보과학회 학술발표논문집 2002년도 봄(B), pp.295-297, 2002.

[5] 서연규, 조성배, "N명 죄수의 딜레마 게임에서 지역화된 상호작용을 통해 진화된 전략의 일반화 능력," 한국정보과학회 학술발표논문집 1999년도 봄(B), pp.230-232, 1999.

[6] Alireza Soltani, Daeyeol Lee and Xiao-Jing Wang, "Neural mechanism for stochastic behaviour during a competitive game," Neural Networks, Vol.19, No.8, pp.1075-1090, 2006.

[7] A. L. Nelson, E. Grant and T. C. Henderson, "Evolution of neural controllers for competitive game playing with teams of mobile robots," Robotics and Autonomous Systems, Vol.46, No.3, pp.135-150, 2004.

[8] 황희수, 진화계산 및 진화디자인, 내하출판사, 2002.

[9] John R. Koza, Genetic Programming, MIT press, 1992.

[10] Geoffrey Miller and Dave Cliff, "Co-evolution of pursuit and evasion I: Biological and game-theoretic foundations," Technical Report CSRP311, 1994.

[11] Stefano Nolfi, Dario Floreano, "Co-evolving predator and prey robots: do 'arm races' arise in artificial evolution," Artificial Life 4, pp.31-335, 1998.

[12] Kam-Chuen Jim, C. Lee Giles, "Talking Helps: Evolving Communicating Agents for the Predator-Prey Pursuit Problem," Artificial Life 6, pp.237-254, 2000.

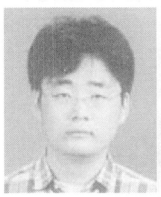
[13] Hitoshi Iba, "Evolutionary learning of communicating agents," Information Sciences, Vol.108, No.1-4, pp.181-205, 1997.

[14] Sreerama K. Murthy, "Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey," Data Mining and Knowledge Discovery, No 2, pp 345-389, 1997.

[15] George H. John, Ron Kohavi, Karl Pfleger, "Irrelevant Features and the Subset Selection Problem," International Conference on Machine Learning, pp.121-129, 1994.

[16] John R. Koza, "Concept Formation And Decision Tree Induction Using The Genetic Programming Paradigm," Parallel Problem Solving from Nature Proceedings of 1st Workshop, PPSN 1, Vol.496, No.1-3, pp.124-128, 1991.

[17] Randall S. Sextona, Robert E. Dorseyb and John D. Johnsonb, "Toward global optimization of neural networks: A comparison of the genetic algorithm and backpropagation," Decision Support Systems, Volume 22, Issue 2, February 1998, pp.171-185.



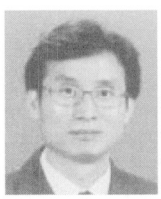
권 오 광

e-mail : fresian@ece.skku.ac.kr

2005년 성균관대학교 전기전자 및 컴퓨터공학부(학사)

2005년~현 재 성균관대학교 컴퓨터공학부 석사과정

관심분야 : 유전 알고리즘, 신경망, 게임 인공지능



박 종 구

e-mail : pjk@skku.ac.kr

1987년 서울대학교 제어계측공학과(학사)

1989년 서울대학교 제어계측공학과 (공학석사)

1993년 서울대학교 제어계측공학과 (공학박사)

1995년~현 재 성균관대학교 정보통신공학부 교수

2002년~현 재 산업자원부 지정 게임기술개발지원센터 센터장

관심분야 : 제어시스템, 가상현실, 게임 알고리즘